STEFAN SCHÖNIG

EIN PROCESS MINING-RAHMENWERK FÜR AGILE, PERSONENBEZOGENE PROZESSE

DISSERTATION

INSTITUT FÜR INFORMATIK
FAKULTÄT FÜR MATHEMATIK, PHYSIK, INFORMATIK

UNIVERSITÄT BAYREUTH

Universität Bayreuth
Institut für Informatik

Ein Process Mining-Rahmenwerk für agile, personenbezogene Prozesse

Von der Universität Bayreuth zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von
Stefan Konrad Alfred Schönig
geboren in Weiden i. d. OPf.

Erster Gutachter: Prof. Dr.-Ing. Stefan Jablonski
Zweiter Gutachter: Prof. Dr. Michael zur Muehlen
Tag der Einreichung: 15. Juli 2015
Tag des Kolloquiums: 26. November 2015

KURZBESCHREIBUNG

Prozessmanagement gilt als Ansatz, durch Modellierung und Unterstützung von Prozessen in Organisationen, Dienstleistungen effektiv zu gestalten. Es existieren verschiedene Arten von Prozessen: Strikte Prozesse, deren wiederkehrende Abläufe bereits zur Entwurfszeit festgelegt werden können und agile, personenbezogene Prozesse, deren Verhalten sich nicht vollständig im Voraus festlegen lässt und die auf menschlichen Entscheidungen basieren. Diese Art von Prozessen findet sich in allen Bereichen der Gesellschaft wieder, beispielsweise in Forschungs- und Entwicklungsprozessen oder im Gesundheitswesen. Zur Modellierung von Prozessen existieren zwei Ansätze: Prozedurale und regelbasierte Modellierungssprachen. Während die prozedurale Modellierung eher für strikte Prozesse geeignet ist, können agile, personenbezogene Prozesse besser durch regelbasierte Modelle beschrieben werden. Ein regelbasiertes Prozessmodell umfasst die Definition beteiligter Entitäten und eine Menge von Regeln. Abläufe müssen nicht explizit modelliert werden, sondern ergeben sich automatisch durch Berücksichtigung der Regeln.

Process Discovery bezeichnet das Sammeln von Informationen über einen Prozess und deren strukturierte Abbildung in einem Prozessmodell. Die Akquisition des Prozesswissens ist ein aufwändiges Unterfangen. Ein Ansatz zur Unterstützung und (Teil-)Automatisierung von Process Discovery ist Process Mining. Process Mining-Techniken verwenden Ausführungsinformationen, welche digital in Datenbanken oder Dateien vorliegen, um daraus automatisiert Prozessmodelle abzuleiten. Klassische Process Mining-Techniken erzeugen Prozessmodelle in prozeduralen Modellierungssprachen, die für agile Prozesse eher ungeeignet sind. In den letzten Jahren wurden einige Techniken zur Ableitung von regelbasierten Prozessmodellen aus Ereignisprotokollen vorgeschlagen. Der Fokus all dieser Arbeiten liegt auf der Analyse der Ausführungsreihenfolge von Aktivitäten, mit einigen Erweiterungen für Daten. Prozessausführende Akteure und deren tiefgreifender Einfluss auf den Prozessablauf werden in all diesen Arbeiten nicht betrachtet. Ansätze, welche diese organisatorische Perspektive von Prozessen analysieren, wurden bislang nicht in regelbasierte Mining-Verfahren integriert und sind daher für agile Prozesse ungeeignet.

Die Forschung dieser Dissertation vereinigt diese beiden Strömungen und beschreibt ein Process Mining-Verfahren zur Ableitung von regelbasierten, perspektivenübergreifenden Prozessmodellen. Die abgeleiteten Modelle beschreiben nicht nur komplexe organisatorische Zuweisungsregeln sondern geben außerdem Einblick in den tiefgreifenden Einfluss von Personen auf die Ausführungsreihenfolge von Aktivitäten. Durch Anwendung des vorgestellten Ansatzes auf Ereignisprotokolle agiler, personenbezogener Prozesse, werden sowohl Funktionalität als auch Anwendbarkeit und Effizienz des Verfahrens belegt.

ABSTRACT

The success of an organisation primarily depends upon its ability to accomplish its tasks in a structured and reliable manner. A well accepted method for structuring the activities carried out in an organisation is business process management (BPM). BPM usually involves modelling, executing and analysing processes. In this context, two different types of processes can be distinguished: well-structured routine processes with exactly predescribed control flow and agile processes with control flow that evolves at run time without being fully predefined a priori. Agile processes are common in healthcare where, e.g., patient diagnosis and treatment processes require flexibility to cope with unanticipated circumstances. In a similar way, two different representational paradigms can be distinguished: procedural models describe which activities can be executed next and declarative models define execution constraints that the process has to satisfy. The more constraints are added to the model, the less possible execution alternatives remain. As agile processes may not be completely known a priori, they can often be captured more easily using a declarative rather than a procedural modelling approach.

For purposes of compliance and process improvement, organisations are interested in the way their processes are actually executed. Since process execution is often supported by information systems, process events such as starts and completions of activities are recorded in so-called event logs. Process mining aims at discovering processes by extracting knowledge from event logs, e.g., by generating a process model reflecting the behaviour recorded in the logs. Declarative languages like Declare or DPIL can be used to represent these models, and tools like the DeclareMiner or MINERful to generate them, often with a focus on control flow. Agile processes, however, need to explicitly integrate the organisational perspective due to the importance of human decision-making. Recent research has identified the potential of process mining of the organisational perspective. However, these results have not yet been integrated with declarative process models and are therefore not suitable in agile environments.

This thesis fills this research gap by proposing a process mining approach to discover rule-based process models that comprise complex organizational patterns. In particular, we are able to extract complex allocation rules, such as binding of duties between activities, as well as cross-perspective rules involving the control-flow and the organisational perspectives together. The latter consider the influence of resource allocation on the control flow of the process, e.g., an activity can only be executed by a specific role if a specific activity was performed previously. The expressiveness of the extracted rules has been evaluated using the Workflow Resource Patterns. The applicability of the approach has been validated with real-world event logs from different domains.

DANKSAGUNG

Diese Dissertation wäre wohl niemals ohne die Unterstützung von vielen netten Menschen entstanden, bei denen ich mich an dieser Stelle von ganzem Herzen bedanken möchte.

Allen voran gilt mein Dank meinem Doktorvater Prof. Dr.-Ing. Stefan Jablonski, der mir durch das entgegengebrachte Vertrauen die Chance zur Promotion eröffnet hat. Während meiner Arbeit am Lehrstuhl stand er mir stets mit Rat und Tat zur Seite und hat durch zahlreiche Anregungen, Hinweise und Diskussionen richtungsweisend zum vorliegenden Resultat beigetragen. Durch seine Offenheit und Herzlichkeit schuf er ein äußerst harmonisches Arbeitsklima, für das ich ihm ebenfalls sehr dankbar bin.

Ganz herzlich bedanken möchte ich mich auch bei all meinen Kollegen für die zahlreichen fachlichen Gespräche sowie hilfreiche und konstruktive Kritik. Namentlich hervorheben möchte ich Lars Ackermann, Florian Gillitzer, Matthias Jahn, Raimund Matros, Bastian Roth und Michael Zeising, zu denen während meiner Zeit am Lehrstuhl auch ein freundschaftliches Verhältnis entstanden ist. Besonderer Dank gilt Michael. Zusammen haben wir in den letzten Jahren zahlreiche Publikationen verfasst und uns gegenseitig bei Forschung und Entwicklung von Systemen unterstützt. Weiterer Dank gebührt Prof. Dr. Michael zur Mühlen für die Übernahme des Zweitgutachtens sowie seine wertvollen Anmerkungen während der Diskussionen bei unserem Treffen in New York. Gleichermaßen möchte ich unseren Sekretärinnen Christine Leinberger und Kerstin Haseloff sowie unserem Techniker Bernd Schlesier für den reibungslosen Ablauf am Lehrstuhl und für die netten und ermunternden Gespräche danken.

Ein riesengroßes Dankeschön möchte ich meiner Familie, allen voran meinen Eltern und meinem Bruder Tobias aussprechen, die mich auf meinem gesamten Lebensweg bedingungslos unterstützt und mir stets neue Kraft und Motivation gegeben haben. Ohne euch hätte ich wohl nie diesen Weg eingeschlagen. Zu guter Letzt möchte ich Cristina Cabanillas von ganzem Herzen danken. Sie war stets eine verständnisvolle Zuhörerin und hat mich zudem mit ihrem fachlichen Wissen bei der Fertigstellung dieser Arbeit unterstützt - Muchas gracias, Cristina. Nur durch eure Unterstützung konnte ich diesen langen und schweren Weg meistern - Vielen Dank für Alles!

Diese Promotion wurde im Rahmen des Projektes “Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement” (KpPQ) durchgeführt, das vom Europäischen Fonds für regionale Entwicklung (EFRE, Fördernummer 1502/89304-01/2012) gefördert wurde.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

1 EINFÜHRUNG UND PROBLEMSTELLUNG

INHALT

Strukturierte Herangehensweise an Aufgaben

In einer weltweit vernetzten Welt sind Menschen und Organisationen kontinuierlich mit einer Vielzahl an Aufgaben und Aktivitäten steigender Komplexität konfrontiert. Eine **strukturierte Herangehensweise** zur Abarbeitung dieser Aufgaben und zur Lösung von Problemen ist daher unerlässlich. Immer mehr Menschen, Organisationen und Unternehmen besitzen das Bedürfnis, Arbeitsabläufe zu strukturieren und detailliert zu dokumentieren, um effizient, fehlerfrei und nachvollziehbar arbeiten zu können. Die Effizienz und Effektivität bei der Entwicklung von Produkten und Dienstleistungen ist entscheidend für den wirtschaftlichen Erfolg eines Unternehmens.

1.1 PROZESSMANAGEMENT UND PROCESS MINING

Prozessmanagement häufig Grundlage für Qualitätsmanagement

Prozessmanagement gilt als Ansatz, durch Identifikation, Modellierung und Unterstützung von Abläufen in Organisationen (d.h. **Prozessen**), Produkte effizient zu entwickeln und Dienstleistungen effektiv zu gestalten [42], [138]. Prozessmanagement ist heute sehr oft notwendige **Grundlage eines Qualitätsmanagements** [49].

1.1.1 Phasen und Prozesslebenszyklus

Phasen des Prozessmanagement

Die Aufgaben und Phasen des Prozessmanagements werden in der Literatur üblicherweise im so genannten **Prozesslebenszyklus** (*engl. process lifecycle*) zusammengefasst und erläutert. Es existieren verschiedene Definition dieses Lebenszyklus. In dieser Arbeit wird der Ansatz aus [47] verwendet. Abbildung 1 gibt einen Überblick über die verschiedenen Phasen des Prozessmanagements und die Artefakte, die zwischen diesen Phasen fließen.

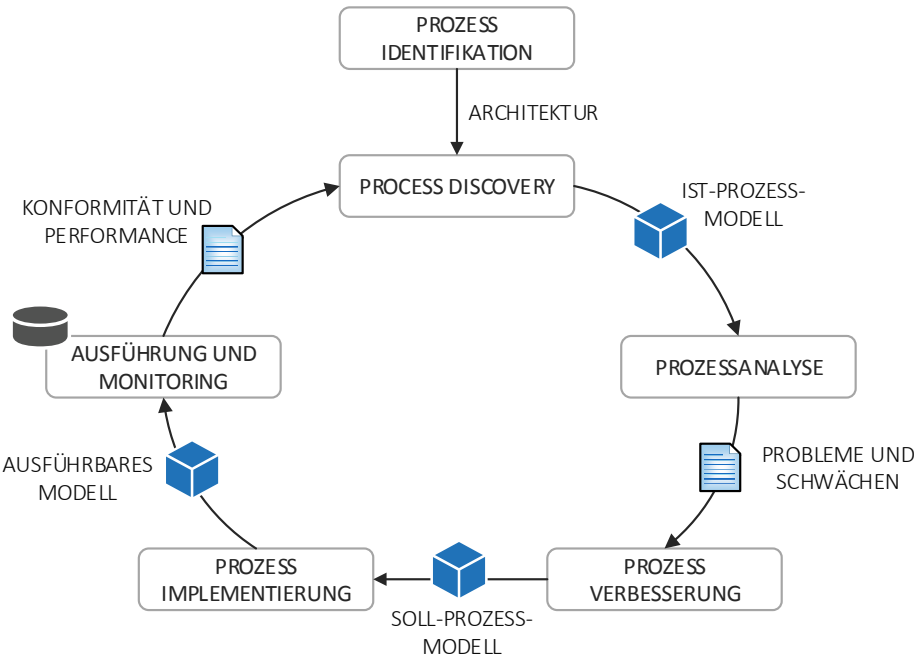


Abb. 1: Phasen und Artefakte des Processlebenszyklus

Identifikation

In der **Identifikationsphase** (*engl. process identification*) werden die Ziele des Prozessmanagements in einer Organisation und die relevanten Prozesse definiert. Das Ergebnis der Identifikation ist eine neue oder aktualisierte Prozessarchitektur, die einen Überblick über die zu unterstützenden Prozesse gibt. Außerdem werden die formalen und inhaltlichen Anforderungen an diese Prozesse, deren Modellierung und Ausführung definiert und wie diese Anforderungen umgesetzt werden sollen [87]. Das Ergebnis ist die Entscheidung für eine Prozessmodellierungssprache und ein Werkzeug zur Unterstützung der Ausführung.

Process Discovery

In der **Process Discovery**-Phase[1], auch **Ist-Prozessmodellierung** genannt, werden die relevanten, aktuell ablaufenden Prozesse der betrachteten Organisation dokumentiert und modelliert. Zur Modellierung von Prozessen wurden im Laufe der Zeit entsprechende Prozessmodellierungssprachen entwickelt. Eine derartige Sprache umfasst Entitäten aus denen das Modell zusammengesetzt ist, z.B. Aktivitäten und Akteure, sowie Regeln, nach denen diese Entitäten kombiniert werden dürfen (*Syntax*). Außerdem definiert sie die Bedeutung der Entitäten sowie deren Kombinationen (*Semantik*). Die definierte Sprache bzw. Notation wird hier verwendet, um Prozesse (d.h. Aktivitäten, Abläufe und deren Abhängigkeiten zu beteiligten Entitäten wie Akteuren) in einem **Ist-Prozessmodell** abzubilden. Diese Tätigkeit verlangt tiefgreifendes Fachwissen der zu Grunde liegenden Domäne, um alle verschiedenen Bereiche eines Prozesses adäquat beleuchten zu können [7].

Modellierungssprachen

Ist-Prozessmodell

Analyse

In der **Analysephase** (*engl. process analysis*) werden anhand der Ist-Modelle Probleme und Schwächen der aktuellen Prozesse identifiziert und dokumentiert. Das Ergebnis dieser Phase ist typischerweise eine strukturierte Liste an Problemen.

Verbesserung

Das Ziel der **Verbesserungsphase** (*engl. process redesign, process improvement*) ist es, die bestehenden Prozesse so zu verändern, dass die identifizierten Probleme und Schwächen behoben werden. Das Ergebnis dieser Phase ist ein veränderter, verbesserter Prozess, abgebildet in einem sogenannter **Soll-Prozessmodell**.

Soll-Prozessmodell

Implementierung bzw. Umsetzung

Auf diese Weise definierte Soll-Prozessmodelle werden in der **Implementierungsphase** (*engl. process implementation*) umgesetzt und in der Organisation realisiert. Hier müssen typischerweise zwei Aspekte betrachtet werden: Einerseits die Umsetzung der Prozessunterstützung (*engl. process automation, process execution support*), andererseits die Organisationsveränderung (*engl. organizational change management*). Organisationsveränderung bezieht sich auf alle Aktivitäten, die notwendig sind, um die Arbeitsweise der beteiligten Akteure auf Basis der Prozessunterstützung anzupassen, z.B. durch Verwendung eines IT-Systems oder einer Checkliste. Die Umsetzung der Prozessunterstützung beschreibt hingegen die Anpassung und Einrichtung von Systemen, auf Basis derer die Durchführung des definierten Soll-Prozesses unterstützt wird. Prozessmodelle können sowohl auf Basis IT-gestützter Informationssysteme als auch ohne IT-Unterstützung verwendet werden. Bei erstgenannter Verwendung erfolgt der Einsatz eines **Prozessausführungssystems** (PAS), zur automatisierten Ausführung der Prozessmodelle bzw. der darin definierten Aktivitäten (*engl. process automation*). PAS interpretieren Prozessmodelle und weisen darauf basierend Aufgaben und Informationen an andere Systeme, Dienste und Anwendungen (*Orchestrierung*) oder direkt an menschliche Prozessteilnehmer zu. PAS unterstützen auf diese Weise die Kollaboration und Kommunikation zwischen prozessbeteiligten Anwendern durch die Bereitstellung spezifischer Information zur rechten Zeit [106]. Neben der IT-gestützten Ausführung besteht auch die Möglichkeit, Prozesse ohne IT-basierte Entscheidungsunter-

Prozessausführungssystem

1 Da sich der Begriff "Process Discovery" auch im nicht-englischsprachigen Raum etabliert hat, wird in dieser Arbeit auf eine Übersetzung verzichtet.

stützung, beispielsweise durch Verwendung von prozessbasierten Checklisten, durchzuführen [23], [115]. Häufg spricht man bei der Prozessausführung ohne IT-Unterstützung auch von **manueller** Ausführung. Prozessmodelle dienen hier zur Dokumentation, Nachvollziehbarkeit und zur Kommunikation zwischen beteiligten Mitarbeitern.

Ausführung und Monitoring

In der **Ausführungs- und Monitoringphase** (*engl. process execution, monitoring*) werden die definierten Prozessmodelle schließlich, unterstützt durch PAS, im Unternehmen "gelebt", d.h. ausgeführt. Parallel zur Ausführung von Prozessen werden unterschiedliche Daten in Form von Ausführungsprotokollen (*engl. Logs*) erzeugt bzw. aufgezeichnet (engl. *Logging*), welche den tatsächlichen Prozessablauf widerspiegeln. Bei der IT-basierten Ausführung von Prozessen werden Informationen über Prozessereignisse (z.B. Start und Beendigung von Aktivitäten), deren zeitliche Reihenfolge sowie häufig auch Daten zu ausführenden Akteuren automatisiert aufgezeichnet [11]. Die auf diese Weise erzeugten Daten werden analysiert, um Unvollständigkeiten in Prozessmodellen, Fehler, Abweichungen oder Ineffizienzen zu identifizieren. Es entstehen potentiell neue Probleme, die eine erneute Aufnahme, Analyse und Verbesserung der betrachteten Prozess erfordern. Auf diese Weise schließt sich der Prozesslebenszyklus und der Prozess entwickelt sich stetig weiter (engl. evolution).

Logging von Ereignissen

Evolution

1.1.2 Process Discovery und Process Mining

Sammeln von Information und Abbildung

Process Discovery[2] bezeichnet das Sammeln von Informationen über einen Prozess und deren strukturierte Abbildung in einem Ist-Prozessmodell [47]. Die Modellierungstätigkeit kann dabei erst beginnen, wenn genügend Informationen zusammengetragen wurden. Die Akquisition des Prozesswissens ist ein aufwändiges Unterfangen [7], [47]. Grund dafür ist vor allem, dass Prozesswissen, insbesondere in größeren Unternehmen, in der Regel auf viele Beteiligte verteilt ist, die jeweils über spezifisches Prozesswissen über ihre Ressort verfügen [47]. Die Ist-Prozessmodellierung ist zentral und ausschlaggebend für ein funktionierendes und effektives Prozessmanagement [87]. Die Aufgabe der Modellierung wird daher vornehmlich von Modellierungsexperten übernommen. Unter Verwendung von traditionellen Techniken, z.B. **Interviews** und **Workshops**, versuchen Prozessmodellierer Prozesswissen zu sammeln und zu globalem Prozesswissen zu aggregieren [47]. Dennoch stellt die Akquisition von Prozesswissen auch für den Fachmann eine große Herausforderung dar, die mit vielen Hindernissen verbunden ist. Aussagen von beteiligten Domänenexperten sind in vielen Fällen **subjektiv** und nicht immer sind Beteiligte auch daran interessiert, ihr Wissen öffentlich zugänglich zu machen. Die Ist-Modellierung der Abläufe in Organisationen ist daher eine langwierige, kostspielige aber dennoch fundamentale und gewinnbringende Aufgabe [47].

Akquisition von Prozesswissen aufwändig

Traditionelle Discovery-Techniken

IT-Unterstützung durch Process Mining

Neben Interviews und Workshops kann die Process Discovery-Phase auch

2 Diese Phase wird in der Literatur häufig auch als "Process Design" bezeichnet. Der Begriff "Discovery" spiegelt jedoch eher die Realität wieder, da der Prozess nicht neu definiert wird, sondern zumindest implizit bereits in den Köpfen der beteiligten Akteure existiert.

systematisch durch IT unterstützt werden [11], [47]. Ein Ansatz, der bereits jetzt erfolgreich in zahlreichen Projekten in Wirtschaft und Industrie Anwendung findet, ist **Process Mining**[3] [11]. Process Mining bezeichnet eine Gruppe von Verfahren, welche Ideen und Methoden aus den Bereichen **Data Mining** (Muster- und Wissensextraktion aus großen Datenbeständen) und der klassischen Prozessmodellierung kombinieren. Moderne Informationssysteme, wie ERP-Systeme (z.B. SAP) und CRM-Systeme (z.B. SalesForce), zeichnen eine Vielzahl an Ereignissen wie Benutzeraktivitäten und Interaktionen digital auf. Process Mining-Techniken verwenden diese Ausführungsinformationen, welche digital in Datenbanken oder Dateien vorliegen, um daraus **automatisiert** Prozessmodelle abzuleiten. Modelle, die mit Process Mining gewonnen werden, sind anders als Modelle, die mittels herkömmlichen Techniken ermittelt werden, **objektiv** und spiegeln so den Prozess wieder, wie er wirklich ausgeführt wurde (*engl. evidence-based process discovery*). Durch die Analyse des "digitalen Fußabdrucks" kann Process Mining die Ist-Modellierung von Prozessen (teil-) automatisieren und Einblicke in die tatsächlich ausgeführten Prozesse gewähren. Es existieren Verfahren zur (automatisierten) **Modellierung** (d.h. Rekonstruktion von Prozessmodellen aus Ereignisprotokollen), **Konformitätsprüfung** (d.h. Erkennung von Abweichungen durch Vergleich bestehender Modelle mit Ereignisprotokollen) und **Erweiterung** (d.h. Vervollständigung bestehender Modelle auf Basis von Ereignisprotokollen) von Prozessen. Ausgangspunkt für den erfolgreichen Einsatz von Process Mining ist ein digital zugängliches, konsistentes und möglichst vollständiges Ereignisprotokoll historischer Prozessabläufe, d.h. bereits durchgeführter *Prozessinstanzen* [11].

Objektive Prozessmodelle

Typen von Process Mining-Verfahren

1.1.3 Perspektiven der Prozessmodellierung

Zur systematischen Heransgehensweise bei der Prozessmodellierung und zur Strukturierung von Prozessen werden grundlegende Perspektiven identifiziert [40]. Die **funktionale Perspektive** beschreibt die funktionalen Bausteine eines Prozesses. Elementare Prozessschritte bilden die kleinsten Arbeitseinheiten und erfordern üblicherweise personelle und technische Ressourcen. Im ersten Fall werden sie während der Ausführung des Prozesses den Prozessteilnehmern zugewiesen. Zusammengesetzte Prozessschritte hingegen verweisen auf ein untergeordnetes Prozessmodell. In diesem Fall wird der Prozessschritt durch das untergeordnete Prozessmodell genauer definiert. Zusammengesetzte Prozessschritte ermöglichen die Wiederverwendung von Teilprozessen und die Strukturierung von umfangreichen Prozessmodellen. Die **verhaltensorientierte Perspektive** beschreibt das zeitliche Verhalten eines Prozesses und wird häufig auch als Kontrollfluss bezeichnet. Die **organisatorische Perspektive** beschreibt die Zuweisung von Prozessschritten zu menschlichen Prozessteilnehmern, d.h. Akteuren. Zu diesem Zweck referenzieren Prozesse üblicherweise ein organisatorisches Modell, das Personen, organisatorische Einheiten und Beziehungen

Funktionale Perspektive

Verhaltensorientierte Perspektive

Organisatorische Perspektive

3 Die Begriffe "Process Mining" und "Data Mining" sind feste Begriffe in der Wissenschaftswelt und werden nicht übersetzt.

Daten-Perspektive

Operationale Perspektive

abbildet. Die **datenorientierte Perspektive** beschreibt, welche Daten- bzw. Informationseinheiten an welcher Stelle im Prozess benötigt, erzeugt oder manipuliert werden. Die **operationale Perspektive** beschreibt wie ein atomarer Prozessschritt implementiert ist und welche Applikationen und Werkzeuge zu dessen Durchführung benötigt werden [68]. Je nach Domäne und Anwendungsgebiet können bei Bedarf weitere Perspektiven definiert und beschrieben werden.

Ortsbezogene Perspektive

Ein Beispiel ist die **ortsbezogene Perspektive**, welche beispielsweise die Abhängigkeiten der Prozessdurchführung von Mitarbeiterpositionen und Standorten beschreibt [122]. Durch eine ganzheitliche Betrachtung aller Perspektiven ergibt sich schließlich der resultierende Prozessablauf.

1.2 ARTEN VON PROZESSEN

Arten von Prozessen

Bereits 1994 wird gezeigt, dass in der Realität verschiedene Arten von Prozessen existieren [67], welche sich durch unterschiedliche Eigenschaften und unterschiedliche Anforderungen an die Modellierung auszeichnen [106].

- **Strikte Prozesse**, deren wiederkehrende Abläufe bereits zur Entwurfszeit vollständig festgelegt werden können und
- **Agile, personenbezogene Prozesse**, deren Verhalten sich nicht vollständig im Voraus festlegen lässt und die auf komplexen, menschlichen Entscheidungen basieren.

1.2.1 Strikte Prozesse

Strikte Routineprozesse

Zahlreiche Prozesse in Organisationen bilden die Grundlage für standardisierte Abläufe und können daher als **strikte Routineprozesse** bezeichnet werden. Derartige Prozesse werden häufig auf die gleiche Weise durchgeführt und besitzen eine überschaubare Anzahl an unterschiedlichen Ablaufmöglichkeiten [128].

Wenig Handlungsspielraum und wenige Pfade

Sind menschliche Akteure an diesen Prozessen beteiligt, haben diese relativ wenig Handlungsspielraum, weshalb sämtliche Aktionen und Pfade vorhersehbar sind. Daher lassen sich derartige Prozesse bereits zum Modellierungszeitpunkt nahezu vollständig in einem entsprechenden Prozessmodell spezifizieren [106]. Beispiele für strikte, routineartige Prozesse sind Produktions- oder IT-basierte Verwaltungsprozesse, welche die Orchestrierung von Web Services und Applikationen unterstützen. Modelle, welche diese Prozesse beschreiben, sind daher relativ **variantenarm**, d.h. sie besitzen wenige und genau vorgegebene Pfade.

Nur 20-40% aller Prozesse sind strikt

Aufgrund des einfachen und **standardisierbaren** Ablaufs sind keine zusätzlichen Erklärungen und Empfehlungen notwendig. Tatsächlich lassen sich in der Realität nur ca. 20-40% aller Prozesse als strikt einstufen [128].

1.2.2 Agile, personenbezogene Prozesse

Agile, personenbezogene Prozesse

Prozesse in Organisationen besitzen nicht selten eine erhebliche Dynamik, die sich beispielsweise darin zeigt, dass eine Vielzahl von Abläufen nicht vollständig spezifiziert wird bzw. werden kann [135]. Die Ursache dafür liegt darin, dass sich viele Abläufe durch einen hohen Grad an Variabilität auszeichnen, welcher dazu führt, dass Prozesse häufig auf unterschiedliche Weise durchgeführt werden [8]. Die Ausführung dieser **agilen Prozesse** ist in großem Maße abhängig von menschlichen Akteuren, deren Entscheidungen und Expertenwissen und daher vor allem **personenbezogen** [131]. Akteure besitzen in diesen Prozessen deutlich mehr **Entscheidungsfreiheit** bei der Wahl der durchzuführenden Aktivitäten als auch deren Ausführungsreihenfolge [128]. Die tatsächliche Prozessdurchführung wird von prozessbeteiligten Akteuren in Anbetracht vorliegender Informationen in deren eigenem Ermessen festgelegt. Diese Prozesse sind daher besonders **vielseitig** und basieren auf **komplexeren** Zusammenhängen [106]. Menschliche Akteure und organisatorische Beziehungen (**Organisatorische Perspektive**) stehen im Zentrum des Prozesses und bestimmen in großem Maße dessen Ablauf. Man spricht auch von der Klasse der wissensintensiven Prozesse (*engl. knowledge intensive processes*). Der Forschungsbereich der Fallbehandlung (*engl. case management bzw. case handling*) bezeichnet sie als Fälle (*engl. cases*) [8]. Die Eigenschaften agiler, personenbezogener Prozesse lassen sich folgendermaßen zusammenfassen:

Menschliche Akteure im Vordergrund

Vielseitig und variantenreich

Fallbehandlung (Case Management)

- Akteure besitzen deutlich mehr **Entscheidungsfreiheit** bei der Wahl der durchzuführenden Aktivitäten, als auch deren Ausführungsreihenfolge. Diese Prozesse sind daher **besonders variantenreich**.
- Um beteiligten Akteuren trotz der gegebenen Entscheidungsfreiheit ausreichende Unterstützung zu bieten, ist die **Unterscheidung von verpflichtenden** und lediglich **empfohlenen Aktionen** im Modell sinnvoll [105].
- Sie sind in großem Maße abhängig von menschlichen Akteuren und deren Expertenwissen, d.h. personenbezogen. Aufgaben werden Personen, Rollen oder Benutzergruppen auf Basis **komplexer organisatorischer Zuweisungsregeln** zugewiesen [30].
- Neben der Aufgabenzuweisung hat die organisatorische Perspektive in derartigen Prozessen auch großen **Einfluss auf den Prozessablauf**.

1.2.3 Agiler, personenbezogener Beispielprozess

Agiler Beispielprozess

Zur Erläuterung der Eigenschaften agiler, personenbezogener Prozesse betrachten wir exemplarisch einen vereinfachten Ausschnitt eines universitären Dienstreiseabwicklungsprozesses. Anhand dieses einfachen Beispiels können die soeben beschriebenen Eigenschaften gut aufgezeigt werden.

Zur Abwicklung einer universitären Dienstreise muss einmalig ein Dienstreiseantrag, unter Beteiligung verschiedener Personen und organisatorischer

Rollen, gestellt und anschließend genehmigt werden. Des Weiteren müssen potentiell mehrere Unterkünfte gebucht werden. Anders als bei einem derartigen Prozess zunächst vermutet werden könnte, handelt es sich hierbei um einen agilen und personenbezogenen Prozess. Dies wird durch die folgenden Tatsachen belegt:

- Preise für Unterkünfte ändern sich häufig rapide und kurzfristig. Beispielsweise kann bestimmten Antragstellern im Fall einer Abwesenheit von Verwaltungspersonal die Flexibilität eingeräumt werden, etwaige Buchungen noch vor der Antragsgenehmigung durchzuführen (**Flexibilität, Variantenreichtum**).
- Der Antragsteller muss jedoch darauf hingewiesen werden, dass eine Vorabbuchung ohne Genehmigung eine nicht empfohlene Vorgehensweise ist und auf eigene Verantwortung gehandelt wird (**Empfehlung**).
- Der Antragsteller kennt Umstände, Veranstaltungsorte und eigene Termine am besten. Die Unterkunft muss daher vom Antragsteller selbst gebucht werden (**Verknüpfung von Zuständigkeiten**). Der Antrag muss jedoch auf jeden Fall von einem Mitarbeiter der Verwaltung genehmigt werden (**rollenbasierte Aufgabenzuweisung**).
- Handelt es sich bei dem Antragsteller um einen Professor, der verantwortlich für die entsprechende Kostenstelle ist, so können Unterkünfte noch vor der Antragstellung bzw. Genehmigung gebucht werden. Doktoranden müssen sich an eine strikte Reihenfolge halten. Neben der verhaltensorientierten Perspektive beeinflusst hier demnach auch die organisatorische Perspektive die Ausführungsreihenfolge von Aktivitäten (**Einfluss der organisatorischen Perspektive auf den Prozessablauf**).

Agile Prozesse in allen Bereichen der Gesellschaft

Die zuvor genannten Eigenschaften können bereits anhand dieses einfachen Beispiels aufgezeigt werden. Diese Art von Prozessen findet sich in allen Bereichen der Gesellschaft wieder, beispielsweise in der Versicherungs- oder Kriminalfallbearbeitung, in Forschungs- und Entwicklungsprozessen oder im Gesundheitswesen [8], [128]. Es besteht daher großer Bedarf, die Ist-Modellierung auch dieser Prozesse durch Process Mining-Methoden zu unterstützen. Wie in den folgenden Abschnitten gezeigt wird, kann diese Art von Prozessen mit existierten Process Mining-Verfahren nur unzureichend bearbeitet werden. Ziel dieser Dissertation ist daher die Entwicklung eines **Process Mining-Ansatzes für agile, personenbezogene Prozesse**. Im folgenden Abschnitt werden zunächst die konkreten Anforderungen an ein derartiges Verfahren identifiziert.

Ziel dieser Dissertation

1.3 ANFORDERUNGEN AN PROCESS MINING FÜR AGILE PROZESSE

Anforderungen

Auf Basis der aufgezeigten Eigenschaften lassen sich zunächst zentrale **Anforderungen** an Process Mining im Kontext agiler, personenbezogener Prozesse definieren.

1.3.1 Adäquater Modellierungsansatz

Prozedurale Modellierung

Im Forschungsbereich der Prozessmodellierung existieren aktuell zwei verschiedene Modellierungsansätze bzw. Modellierungsparadigmen [48]. In traditionellen, sog. **prozeduralen** (*engl. procedural*) bzw. imperativen Modellierungssprachen zur Beschreibung von Prozessen wie EPKs [124] und BPMN [96] steht die Abbildung jedes möglichen Ablaufs im Vordergrund. Jede Aktion muss bereits zur Entwurfszeit bekannt sein und in Form von sequentiellen, alternativen und parallelen Pfaden im Modell **explizit** abgebildet werden [48]. Die prozedurale Modellierung ist also für strikte, routineartige Prozesse, bei denen eine relative kleine Anzahl an verschiedenen Abläufen zur Entwurfszeit bekannt ist, besonders gut geeignet [48].

Regelbasierte Modellierung

Die **regelbasierte** (*engl. rule-based*) bzw. deklarative Prozessmodellierung ist ein Ansatz, der besser für die Beschreibung agiler, personenbezogener Prozesse geeignet ist [48], [130]. Ein regelbasiertes Prozessmodell umfasst einerseits die Definition beteiligter Entitäten, andererseits eine Menge von Regeln. Abläufe müssen **nicht explizit** modelliert werden, sondern ergeben sich automatisch durch Berücksichtigung der Regeln und sind somit **implizit** gegeben. Warum wird der regelbasierte Modellierungsansatz den Anforderungen agiler, personenbezogener Prozesse eher gerecht als der prozedurale Ansatz?

- *Alle Aktionen möglich, die keine Regel verletzen*

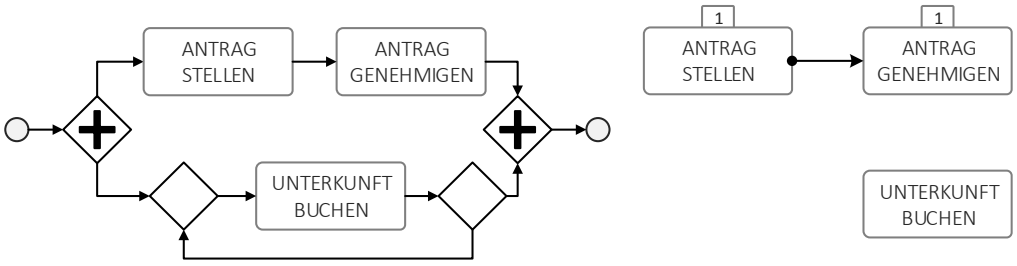
  In einem regelbasierten Modell werden zunächst alle Aktionen als möglich erachtet. Je mehr Regeln zu einem Modell hinzugefügt werden, desto weniger Aktionen sind möglich. Zu einem bestimmten Zeitpunkt sind somit alle Aktionen möglich, die keiner Regel widersprechen. Da nicht jeder mögliche Ablauf explizit modelliert werden muss, sind regelbasierte Modelle zur Abbildung besonders variantenreicher Prozesse gut geeignet [48].

- *Regeln verschiedener Arten*

  Regeln können in verschiedenen Arten definiert werden [106]. Einige Regeln müssen befolgt werden (z.B. gesetzliche Rahmenbedingungen) andere sollten befolgt werden (z.B. Best-Practice). Auf diese Weise kann die in agilen Prozessen benötigte Entscheidungsfreiheit abgebildet werden [105].

- *Abbildung von Beziehungen*

  Regelbasierte Modelle bilden Beziehungen zwischen Entitäten ab und nicht die exakte Ausführungsreihenfolge von Aktivitäten [48]. Sowohl Bedingungen als auch Konsequenzen von Regeln können sich auf sämtliche



(a) Prozedurales Modell **(b)** Regelbasiertes Modell

Abb. 2: Prozedurale und regelbasierte Modelle eines Dienstreiseprozesses

beteiligte Entitäten des Prozesses beziehen, d.h. auf Prozessschritte (Funktion) oder Akteure (Organisation) und deren Verwendung bzw. Ablauf (Verhalten).

Regelbasierte Sprachen

Es existieren verschiedene regelbasierte Prozessmodellierungssprachen. In Abschnitt 3.2 wird deren Eignung als Zielsprache für das vorliegende Process Mining-Verfahren evaluiert und eine adäquate Sprache gewählt. Zur besseren Verständlichkeit und Veranschaulichung verwenden wir in diesem Kapitel eine einfache graphische Hilfsnotation zur Abbildung von regelbasierten Modellen. In Abbildung 2 ist exemplarisch die verhaltensorientierte Perspektive des Dienstreiseprozesses aus Abschnitt 1.2.3 sowohl prozedural (Abbildung 2a) als auch regelbasiert (Abbildung 2b) modelliert.

Verhalten des Prozesses

Im prozeduralen Modell 2a in der BPMN-Notation müssen sämtliche Möglichkeiten explizit im Modell abgebildet sein. Die Nebenläufigkeit bzw. Parallelität zwischen der Antragstellung bzw. dessen Genehmigung und der Buchung der Unterkunft wird durch eine parallele Verzweigung und Vereinigung abgebildet (Raute mit Plus). Da potentiell mehrere Unterkünfte gebucht werden müssen, muss explizit eine Schleife im Modell modelliert werden. Dies wird durch eine exklusive Verzweigung und Vereinigung (Raute) abgebildet. Der Kontrollfluss des Prozesses wird durch insgesamt zehn Sequenz-Flusspfeile modelliert.

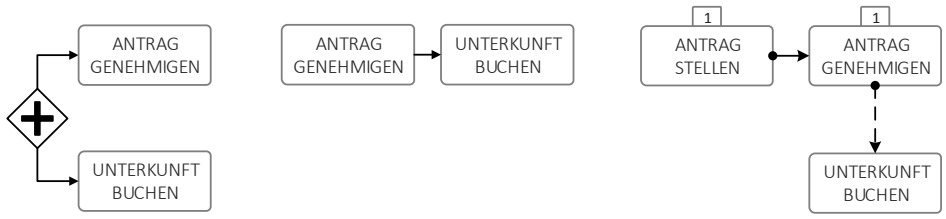
Prozedurales BPMN-Modell

Regelbasiertes Modell

Im regelbasierten Modell (Abbildung 2b) werden lediglich die Regeln bzw. Einschränkungen (*engl. Constraints*) modelliert, denen die Aktivitäten des Prozesses unterliegen. Diese bestehen einerseits darin, dass der Antrag *irgendwann vor* dessen Genehmigung gestellt werden *muss*. Diese zeitliche Abhängigkeit ist exemplarisch durch einen einfachen Pfeil mit Punkt abgebildet. Antragstellung und Genehmigung können nur genau einmal erfolgen, dargestellt durch die Zahl 1 auf diesen Aktivitäten. Für die Buchung der Unterkunft existieren keine zeitlichen Einschränkungen. Sie kann unabhängig von den anderen Aktivitäten und beliebig oft erfolgen. Da der regelbasierte Modellierungsansatz besser zur Abbildung von agilen, personenbezogenen Prozessen geeignet ist ergibt sich Anforderung **A1** an das Verfahren.

Anforderung A1

A1: Abgeleitete agile Prozesse müssen regelbasiert modelliert werden



(a) parallel **(b)** sequentiell **(c)** regelbasiert

Abb. 3: Abbildung von empfohlenen Abläufen in Modellen

1.3.2 Verpflichtende und empfohlene Aktionen

Unterstützung bei Entscheidungsfreiheit

Verschiedene Modalitäten

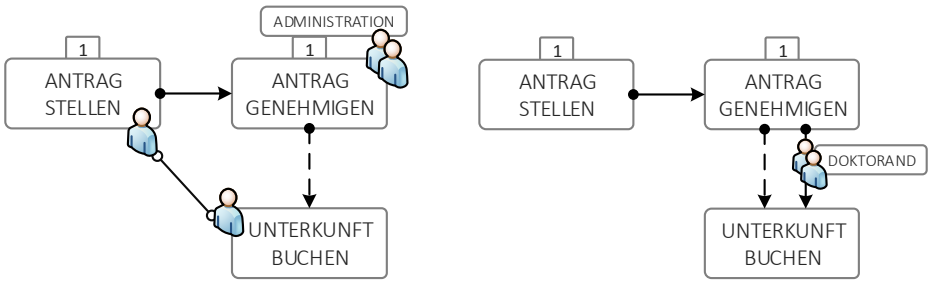
Regelbasierte Modelle eröffnen beteiligten Akteuren große Flexibilität und Entscheidungsfreiheit. Alles was nicht durch Regeln verboten ist, ist erlaubt und daher möglich. Um beteiligten Akteuren trotz der gegebenen Entscheidungsfreiheit ausreichende Unterstützung zu bieten, ist die **Unterscheidung von verpflichtenden und empfohlenen Aktionen** im Modell sinnvoll [20], [105], [106]. Logisch gesehen müssen also zwei *Modalitäten* unterschieden werden können [94]. Dies ist durch die Praxis motiviert. Bei Prozessen im Gesundheitswesen wird beispielsweise zwischen klinischen Richtlinien (verpflichtend) und Leitlinien (empfohlen) unterschieden. Dies lässt sich auf eine Unterscheidung in gesetzlichen Rahmen und bewährtes Vorgehen (*engl. best practice*) verallgemeinern.

Eine Unterscheidung in verpflichtende und empfohlene Aktionen ist zunächst unabhängig vom Modellierungsansatz denkbar. Dennoch bezieht sich diese Anforderung hauptsächlich auf regelbasierte Modelle. Betrachten wir hierzu wieder den Dienstreiseprozess aus Abschnitt 1.2.3. Der Prozessbeschreibung folgend, können bestimmte Antragsteller Unterkünfte bereits vor der Genehmigung des Antrags buchen. Die Einhaltung der bewährten Reihenfolge, d.h. das Abwarten der Genehmigung vor etwaigen Buchungen ist jedoch empfohlen.

Empfohlene Regeln in regelbasierten Modellen

Im regelbasierten Modell in Abbildung 3c ist dieser Zusammenhang durch eine einzige Regel realisiert, nämlich durch die empfohlene zeitliche Abhängigkeit zwischen “Antrag genehmigen” und “Unterkunft buchen”, dargestellt als gestrichelter Pfeil. Ein prozedurales Modell hingegen muss alle möglichen Abläufe explizit enthalten. Modell 3a stellt also alle möglichen Abläufe in der BPMN-Notation dar. Die Aktivitäten sind durch eine parallele Verzweigung voneinander entkoppelt, können also in beliebiger Reihenfolge und nebenläufig zueinander durchgeführt werden. Modell 3b stellt die empfohlenen Abläufe dar. Die beiden Aktivitäten “Antrag genehmigen” und “Unterkunft buchen” sind durch einen Sequenzpfeil verbunden, werden also nacheinander durchgeführt. Derzeit ist nicht klar, wie sich im Fall einer prozeduralen Sprache die beiden Modelle vereinen lassen können und sich die Empfehlung, wie im regelbasierten Fall, durch eine einfache Markierung ausdrücken lässt. Ein Process Mining-Verfahren muss in der Lage sein, bei der Ableitung von Prozessmodellen zwischen verpflichtenden und empfohlenen Regeln zu unterscheiden.

Anforderung A2



(a) Organisatorische Muster **(b)** Perspektivenübergreifendes Muster

Abb. 4: Organisatorische und perspektivenübergreifende Muster im Beispielprozess

A2: Bei der Ableitung von Modellen muss zwischen verpflichtenden und empfohlenen Regeln unterschieden werden

1.3.3 Analyse der organisatorischen Perspektive

Komplexe organisatorische Zuweisungsregeln

Agile, personenbezogene Prozesse sind in großem Maße abhängig von menschlichen Akteuren und Expertenwissen [106], [131]. Aufgaben werden Personen auf Basis **komplexer organisatorischer Zuweisungsregeln** zugewiesen [30]. Ein Process Mining-Verfahren muss daher insbesondere die organisatorische Perspektive des protokollierten Prozesses analysieren. Die entdeckten Muster müssen ebenso in regelbasierte Prozessmodelle integriert werden. Abbildung 4a zeigt ein regelbasiertes Modell des Beispielsprozesses aus Abschnitt 1.2.3, in dem zwei organisatorische Regeln abgebildet sind. Die Linie, welche die beiden Symbole für menschliche Akteure verbindet, stellt die graphische Visualisierung einer Regel zur **Verknüpfung von Zuständigkeiten** dar (*engl. binding of duties*). Unterkünfte müssen demnach vom Antragsteller selbst gebucht werden. Das doppelte Symbol für menschliche Akteure symbolisiert eine **rollenbasierte Aktivitätenzuweisung** (*engl. role based allocation*). Der Antrag muss aufgrund dieser Regel von einem Mitarbeiter der Verwaltungsabteilung genehmigt werden. Diese Regeln stellen Beispiele für organisatorische Zusammenhänge dar, die durch ein Process Mining-Verfahren ableitbar sein müssen. In [111] werden in Form von häufig wiederkehrenden, organisatorischen Mustern in Prozessen (*engl. Workflow Resource Patterns)*, weitere organisatorische Muster definiert, die durch ein Process Mining-Verfahren entdeckt werden sollten. Es ergibt sich Anforderung **A3** an das Verfahren:

Anforderung A3

A3: Das Verfahren muss komplexe organisatorische Muster ableiten und regelbasiert darstellen können

1.3.4 Analyse perspektivenübergreifender Zusammenhänge

Abläufe agiler, personenbezogener Prozesse werden in großem Maße von menschlichen Prozessbeteiligten auf Basis derer Eigenschaften und organisatorischen Position bestimmt [56], [131]. Da für Akteure mit verschiedenen organisatorischen Eigenschaften häufig unterschiedliche Regeln gelten, ergeben sich für verschiedene Personen unterschiedliche Prozessabläufe. Im Beispielprozess müssen beispielsweise nur Doktoranden die Genehmigung des Antrags abwarten, bevor eine Dienstreise gebucht werden kann. Diese Regel ist in Abbildung 4b als Pfeil mit Akteur-Symbol dargestellt. Neben der verhaltensorientierten Perspektive beeinflusst hier demnach auch die organisatorische Rolle eines Akteurs die Ausführungsreihenfolge von Aktivitäten. Während übergreifende Regeln unter Einbeziehung der Verhaltens- und der Datenperspektive in bestehenden Process Mining-Verfahren bereits analysiert werden können, wird eine Kombination von verhaltens- und organisationsbezogenen Aspekten bislang nicht betrachtet. Die operationale Perspektive wird in dieser Arbeit nicht separat behandelt, da verwendete Werkzeuge und Applikationen als konkrete Ressourcen bzw. Akteure betrachtet werden können und daher ähnlich zu analysieren sind.

Operationale Perspektive

Zusammenhänge, welche die verhaltens- und die organisationsorientierte Perspektive gleichzeitig betreffen, bezeichnen wir in dieser Arbeit als **perspektivenübergreifend** [119]. Da sich Regeln sowohl auf die zeitlichen Abhängigen von Aktivitäten (Verhalten) als auch auf Akteure (Organisation) gleichzeitig beziehen können, können diese als **perspektivenübergreifende Modellierungselemente** gut verwendet werden [66]. Perspektivenübergreifende Zusammenhänge finden sich in zahlreichen verschiedenen Anwendungsgebieten wieder, beispielsweise immer in den Situationen, in denen die Durchführung bestimmter Prozessschritte für gewisse Akteure an Vorbedingungen geknüpft ist. Ein häufiges Beispiel aus der Arbeitswelt ist: Während im Allgemeinen keine Überprüfung von Arbeitsergebnissen vor deren Weitergabe erforderlich ist, kann eine Weitergabe *im Fall von Praktikanten* erst nach einer Überprüfung der Ergebnisse erfolgen. Ein weiteres Beispiel aus dem Bereich des Studiums und der Universität ist: Während rückgemeldete Studierende direkt das Studium wiederaufnehmen können, müssen *neu eingeschriebene Studierende* zuvor ein Beratungsgespräch absolvieren. Es ergibt sich Anforderung **A4** an das Process Mining-Verfahren:

Perspektivenübergreifend

Beispiele in Anwendungsgebieten

Anforderung A4

A4: Das Verfahren muss komplexe perspektivenübergreifende Muster ableiten und regelbasiert darstellen können

1.3.5 Effiziente Umsetzung und verständliche Modelle

Orthogonal zu den Anforderungen A1-A4, die sich vorwiegend auf die Ausdrucksstärke abgeleiteter Modelle beziehen, muss ein Process Mining-Verfahren effizient und skalierbar sein und für Analysten verständliche Modelle ableiten.

Effizienz und Skalierbarkeit

Prozessorientierte Ereignisprotokolle können in der Realität sehr umfangreich sein. In vielen Fällen enthalten diese die Ereignisinformation von tausenden Prozessinstanzen und hunderte verschiedene Aktivitäten. Das Ereignisprotokoll eines Krankenhausinformationssystems [28] enthält beispielsweise die Ereignisse von ca. 600 verschiedenen Aktivitäten und mehrere tausend Prozessinstanzen. Um praxistaugliche Laufzeiten zu ermöglichen, muss sich das Verfahren, angesichts dieser Datenmengen, dem Ziel der Analyse entsprechend, konfigurieren lassen und möglichst unnötige Berechnungen vermeiden. Je mehr Elemente ein Modell enthält, desto schwerer fällt es Analysten, den Sinn und die Aussage des Modells zu erfassen [55]. Zur Steigerung der Verständlichkeit extrahierter Modelle muss daher deren Anzahl möglichst minimiert werden. Es ergibt sich Anforderung **A5** an das Verfahren:

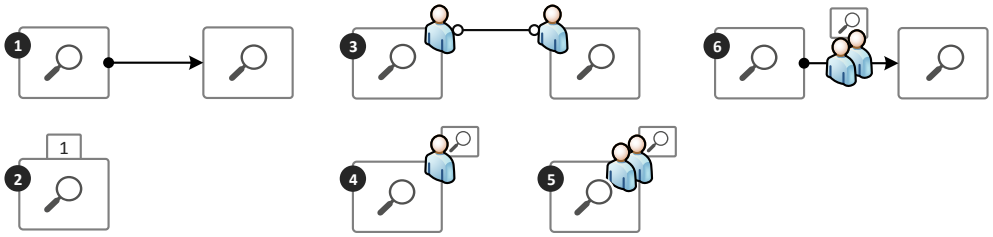
Anforderung A5

A5: Das Verfahren muss effizient sein und verständliche Modelle ableiten

1.4 FORSCHUNGSMETHODIK

Design Science

Die vorliegende Arbeit folgt der Forschungsmethodik des **Design Science** [17]. Im folgenden Abschnitt wird die Forschungsmethodik kurz beschrieben und wie deren Charakteristika im Rahmen dieser Dissertation umgesetzt werden. Die Methodik erweitert die Fähigkeiten von Organisationen, indem sie ein innovatives und zweckmäßiges Artefakt (*Automatisiertes Modellierungs-Verfahren*) für einen relevanten Problembereich (*Agile, personenbezogene Prozesse*) erschafft. Das Artefakt muss einen Nutzen für das Problem bringen, was im Rahmen einer Evaluation (*Implementierung und prototypische Anwendung des Verfahrens*) gezeigt werden muss. Ebenfalls entscheidend ist die Neuartigkeit des Artefakts. Es muss also ein bisher ungelöstes Problem lösen oder ein bestehendes Problem auf effektivere oder effizientere Art lösen als es bisher der Fall ist (*Abgrenzung zu verwandten Arbeiten*). Das Artefakt selbst muss sauber definiert, formal beschrieben, kohärent und in sich konsistent sein (*Formale Beschreibung des Verfahrens*). Schließlich müssen die Ergebnisse der Forschung veröffentlicht werden. Zum einen muss dabei die wissenschaftliche Gemeinschaft adressiert werden (*Wissenschaftliche Publikationen des Autors*), zum anderen muss die Forschung der wirtschaftlichen Gemeinschaft zugänglich gemacht werden, also Wissenschaftlern, die sie im Zusammenhang untersuchen, und Anwender, die sie im eigenen Unternehmen einsetzen (*Anwendung des Verfahrens in Kooperationsprojekten*). Die Methode des Design Science steht im Gegensatz zur Verhaltensforschung, die Theorien zur Erklärung oder zur Vorhersage des Verhaltens von Menschen oder Organisationen entwickelt und verifiziert.



(a) verhaltensorieniert **(b)** organisationsorientiert **(c)** übergreifend

Abb. 5: Exemplarische Regel- bzw. Suchvorlagen zur Ableitung diverser Mustern

1.5 LÖSUNGSANSATZ

Regelbasiertes Process Mining

In dieser Arbeit wird ein Process Mining-Verfahren konzipiert, das anders als viele der existierenden Process Mining-Methoden keine prozeduralen, sondern regelbasierte Prozessmodelle aus Ereignisprotokollen ableitet.

1.5.1 Ableiten regelbasierter Prozessmodelle

Auswahl einer adäquaten Zielsprache

Die Aussagekraft der Resultate von Process Mining ist in großem Maße von der verwendeten **Zielsprache** bzw. Notation abhängig. Nur Muster, die in der gewählten Sprache abgebildet werden können, können auch entdeckt werden [10]. Generelle Voraussetzung ist daher die Auswahl einer Modellierungssprache, mit der sich die in den Anforderungen beschriebenen, komplexen organisatorischen Zusammenhänge abbilden lassen. Kapitel 3 dieser Arbeit beschäftigt sich mit der Evaluation und Auswahl einer adäquaten Modellierungsnotation für agile, personenbezogene Prozesse.

Regelvorlagen

Zur Ableitung von regelbasierten Prozessmodellen aus Ereignisprotokollen werden in der gewählten Sprache sog. **Regelvorlagen** (*engl. rule templates*) definiert. Diese enthalten Platzhalter für konkrete Werte und stellen *Schablonen* der Zusammenhänge dar, die im gegebenen Ereignisprotokoll entdeckt werden sollen. Die Analyse einer bestimmten Regelvorlage kann als **Suchanfrage** (*engl. query*) an das Ereignisprotokoll betrachtet werden. Lösungen der Suchanfrage sind alle Kombinationen von passenden Werten für die Platzhalter, die eine konkrete Regel ergeben, welche im betrachteten Ereignisprotokoll erfüllt ist. Abbildung 5 visualisiert ausgewählte Regelvorlagen und deren Platzhalter in einer einfachen graphischen Notation, auf Basis derer nach den in Abschnitt 1.3 exemplarisch erläuterten Zusammenhängen gesucht werden kann.

Analyse einer Regelvorlage als Suchanfrage

Exemplarische Regelvorlagen der verhaltensorientierten Perspektive

In Abbildung 5a sind Regelvorlagen dargestellt, auf Basis derer Muster der verhaltensorientierten Perspektive abgeleitet werden. Mit Regelvorlage (1) wird beispielsweise nach allen Kombinationen von Aktivitäten gesucht, zwischen denen eine zeitliche Abhängigkeit besteht. Lösungen für diese Suchanfrage sind somit alle Aktivitätenpaare, in denen eine Aktivität beendet sein muss, bevor die andere gestartet werden kann (“*A vor B*”). Analog dazu werden durch Analyse von Regelvorlage (2) alle diejenigen Aktivitäten abgeleitet, die in jeder Pro-

zessinstanz höchstens einmal durchgeführt werden. Abbildung 5b stellt ausgewählte Regelvorlagen der organisatorischen Perspektive dar. Durch Vorlage (3) werden alle Kombinationen von Aktivitäten abgeleitet, die vom gleichen Akteur durchgeführt werden. Vorlage (4) und (5) enthalten je einen Platzhalter für Aktivitäten sowie einen für konkrete Akteure (4) bzw. Gruppen von Akteuren (5). Wird das Protokoll nach diesen beiden Zusammenhängen durchsucht, werden direkte bzw. gruppenbasierte Zuweisungsmuster abgeleitet. Durch Analyse der perspektivenübergreifenden Regelvorlage (6) in Abbildung 5c wird nach allen Kombinationen von Aktivitäten gesucht, zwischen denen nur für Akteure in einer bestimmten organisatorischen Gruppe eine zeitliche Abhängigkeit besteht.

Vorlagen der organisatorischen Perspektive

Perspektivenübergreifende Vorlage

Auf diese Weise definiert die Menge an spezifizierten Regelvorlagen den Suchraum und legt fest, nach welchen Zusammenhängen und Mustern im gegebenen Ereignisprotokoll gesucht wird. Dieser Such- bzw. Analyse-Prozess gliedert sich in vier Teile:

Analyseprozess

- **Identifikation** beteiligter Entitäten: Aus den gegebenen Datenquellen (Ereignisprotokoll, Organisationsmodell) werden die prozessbeteiligten Entitäten, d.h. Aktivitäten, beteiligte Personen und Gruppen, extrahiert.
- **Generierung** von Regelkandidaten: Die spezifizierten Regelvorlagen werden mit **allen möglichen** Kombinationen der identifizierten Entitäten instanziiert. Das Ergebnis ist eine Menge an **Regelkandidaten**.
- **Überprüfung** der Regelkandidaten im Protokoll: Der eigentliche Mining-Prozess - durch Analyse des Ereignisprotokolls wird untersucht, welche Regelkandidaten in den aufgezeichneten Prozessinstanzen erfüllt und welche verletzt sind.
- **Klassifikation** der Kandidaten: Durch eine anschießende Klassifikation werden Regelkandidaten in das resultierende Modell übernommen oder verworfen.

Zur Erläuterung des Verfahrens betrachten wir einen exemplarischen Ausschnitt eines Ereignisprotokolls des Beispielprozesses aus Abschnitt 1.2.3. Tab. 1 stellt die Ereignisse, die zugehörige Aktivität und deren durchführenden Akteur von drei Dienstreisefällen (*Instanzen*) dar. In Prozessinstanz "SS-Riga2013" wird beispielsweise von Akteur "SS" ein Antrag gestellt und, nach dessen Genehmigung durch Person "DS", wiederum durch "SS" eine Unterkunft gebucht. In Instanz "SJ-Muc2014" wird hingegen zuerst eine Unterkunft gebucht bevor ein Antrag gestellt und genehmigt wird.

Ausschnitt eines Ereignisprotokoll

Abbildung 6 visualisiert die Analyse dieses Logs anhand zweier exemplarischer Regelvorlagen. Einerseits wird nach allen Aktivitätenpaaren gesucht, die in einer zeitlichen Abhängigkeit stehen (*"A vor B"*), andererseits nach direkten Zuweisungsregeln, d.h. Aktivitäten, die von bestimmten Akteuren durchgeführt werden (*"A von P"*). Das gesamte Verfahren wird detailliert in Kapitel 4 beschrieben.

Analyse des Beispiel-Logs

Tabelle 1: Exemplarischer Ausschnitt eines Ereignisprotokolls

Fall	Zeitstempel	Ereignistyp	Aktivität	Res.	...
SS-Riga2013	2013-07-06T15:12	Start	Antrag stellen	SS	...
	2013-07-06T15:31	Complete	Antrag stellen	SS	...
	2013-07-09T10:22	Start	Antrag genehmigen	DS	...
	2013-07-09T10:28	Complete	Antrag genehmigen	DS	...
	2013-07-10T09:02	Start	Unterkunft buchen	SS	...
	2013-07-10T09:45	Complete	Unterkunft buchen	SS	...
MZ-Muc2013	2013-10-07T10:07	Start	Antrag stellen	MZ	...
	2013-10-07T10:12	Complete	Antrag stellen	MZ	...
	2013-10-10T08:32	Start	Antrag genehmigen	DS	...
	2013-10-10T10:13	Complete	Antrag genehmigen	DS	...
	2013-10-11T08:06	Start	Unterkunft buchen	MZ	...
	2013-10-11T08:35	Complete	Unterkunft buchen	MZ	...
SJ-Muc2014	2014-11-06T14:58	Start	Unterkunft buchen	SJ	...
	2014-11-06T15:06	Complete	Unterkunft buchen	SJ	...
	2014-11-08T15:36	Start	Antrag stellen	SJ	...
	2014-11-08T15:59	Complete	Antrag stellen	SJ	...
	2014-11-16T10:11	Start	Antrag genehmigen	DS	...
	2014-11-16T10:24	Complete	Antrag genehmigen	DS	...
...	...	...	...	...	...

Im ersten Schritt (*Identifikation*) werden alle Entitäten im Ereignisprotokoll identifiziert, für die freie Platzhalter in den Regelvorlagen existieren. Im Beispiel sind dies alle auftretenden Aktivitäten (*Antrag stellen (S)*, *Antrag genehmigen (G)*, *Unterkunft buchen (B)*) und alle Akteure (*SS*, *MZ*, *SJ*, *DS*). Im zweiten Schritt (*Generierung*) werden alle möglichen Kombinationen in die freien Platzhalter der Regelvorlagen eingesetzt. Auf diese Weise entsteht eine Menge von konkreten Regeln, sog. **Regelkandidaten**. Da im Beispiel-Log drei verschiedene Aktivitäten auftreten und Regelvorlage (1) zwei Platzhalter für Aktivitäten besitzt, werden auf diese Weise 3^2 Kandidaten generiert. Regelvorlage (2) enthält einen Platzhalter Aktivitäten und einen für Akteure, so dass diese Vorlage $3 \cdot 4$ Kandidaten liefert. Im dritten Schritt (*Überprüfung*) werden diese Kandidaten überprüft, d.h. analysiert, in wievielen aufgezeichneten Instanzen des Logs diese erfüllt bzw. verletzt sind. Im Beispiel-Protokoll ist der Regelkandidat *"S vor G"* z.B. zu 100% erfüllt, da stets ein Antrag gestellt wird, bevor dieser genehmigt wird. Die Regel *"G vor B"* ist lediglich in 66% der Instanzen erfüllt, da in Instanz "SJ-Muc2014" zuerst die Buchung und dann die Genehmigung erfolgt. Ferner werden Anträge stets von Akteur "DS" genehmigt, so dass *"G von DS"* zu 100% erfüllt ist. Anhand dieser Werte werden die Regelkandidaten schließlich klassifiziert und darauf basierend als Regeln in das Modell übernommen oder verworfen. Kandidaten, die nahezu immer erfüllt sind, werden als verpflichtende Regeln in das Modell übernommen. Im Beispiel ist dies für *"S vor G"* und *"G von DS"* der Fall. Andere Kandidaten sind zwar nicht immer, jedoch tendenziell erfüllt. Diese Muster werden als empfohlene Regeln in das

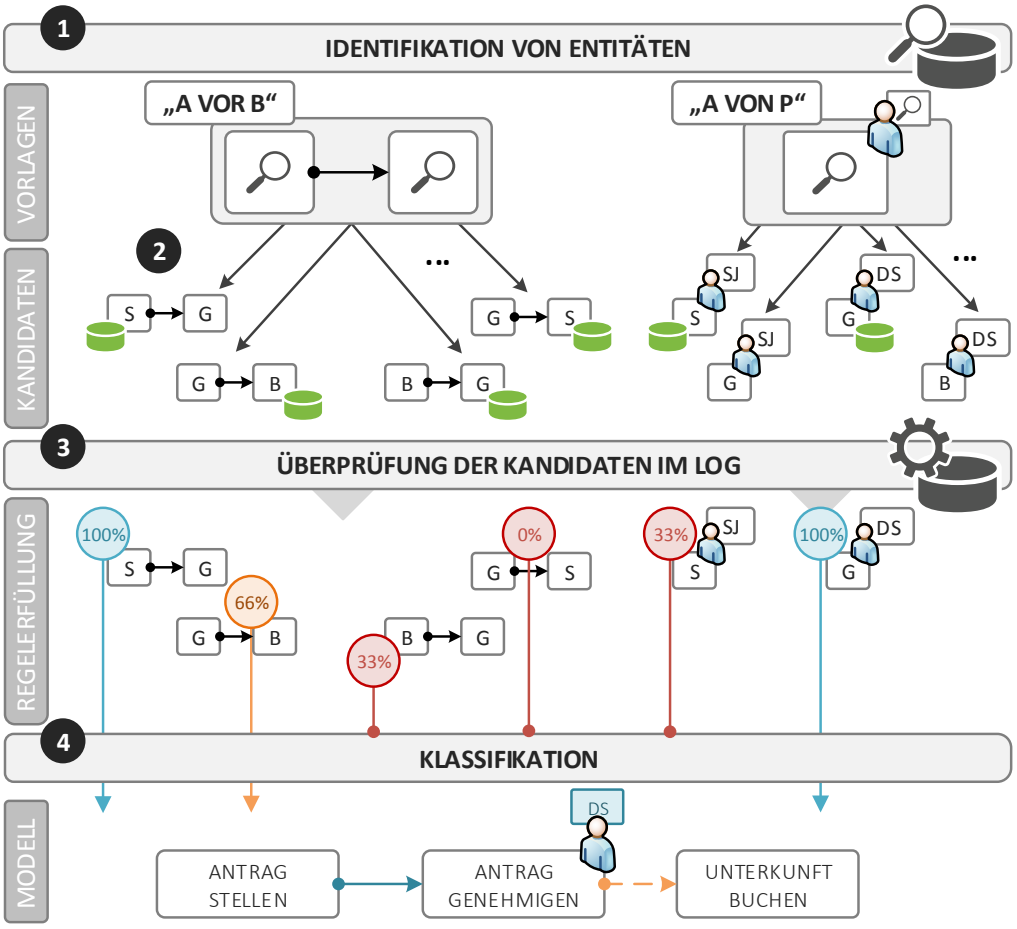


Abb. 6: Beschreibung des Ansatzes zur Ableitung regelbasierter Prozessmodelle

Modell übernommen. Im Beispiel sollte eine Genehmigung vor Buchung der Unterkunft vorliegen, muss jedoch nicht. Auf diese Weise werden Regeln unterschiedlicher Modalität extrahiert. Alle anderen Kandidaten sind im überwiegenden Teil der Instanzen verletzt und daher nicht im resultierenden Modell.

1.5.2 Voranalyse von Ereignisprotokollen

Aufwändige Regelüberprüfung

Der aufwändigste und rechenintensivste Schritt im Analyseprozess besteht in der **Überprüfung von Regelkandidaten** in den aufgezeichneten Prozessinstanzen. Hier sollten unnötige Berechnungen möglichst eingeschränkt werden. Hierzu betrachten wir exemplarisch die Menge an Regelkandidaten in Abbildung 6. Viele der erzeugten Kombinationen treten nie zusammen in einer Prozessinstanz auf. Beispielsweise macht es keinen Sinn die Regelkandidaten *“S von DS”* oder *“B von DS”* zu erzeugen und zu überprüfen, da ein Antrag nie von “DS” gestellt und eine Unterkunft nie von “DS” gebucht wurde. Die Erzeugung und Überprüfung aller möglichen Kombinationen führt daher zu unnötiger Komplexität. Stattdessen ist eine intelligente Erzeugung von Regelkandidaten sinn-

Sinnlose Regelkandidaten vermeiden

voll. Durch eine effiziente Voranalyse des Logs durch das Ableiten häufiger Muster *(engl. frequent pattern mining)*, werden sinnlose Kombinationen vorgefiltert und anschließend nur für tatsächlich zusammen auftretende Entitäten Regelkandidaten erzeugt. Diese sind in Abbildung 6 durch ein Datenbanksymbol gekennzeichnet. Im Beispiel kann auf diese Weise für Regelvorlage (4) (*"A von P"*) die Anzahl an Regelkandidaten von 12 auf 7 zu überprüfende Regeln reduziert und daher nahezu halbiert werden.

1.5.3 Nachbearbeitung von abgeleiteten Modellen

Redundante Regeln

Die Analyse von Ereignisprotokollen auf Basis einer Menge von Regelvorlagen liefert in vielen Fällen Modelle, die *redundante*, d.h. überflüssige Regeln enthalten. Zur Steigerung der Verständlichkeit extrahierter Modelle muss deren Anzahl logischerweise möglichst minimiert werden. Betrachten wir hierzu exemplarisch das Modell in Abbildung 7a, das durch Analyse des Beispiel-Logs auf Basis aller Regelvorlagen aus Abbildung 5 abgeleitet wurde. Bereits dieses einfache Modell enthält einige Redundanzen. Exemplarisch seien hier herausgestellt:

- Vorausgesetzt Akteur "DS" sei Mitglied in der organisatorischen Gruppe "Administration". Durch Analyse von Vorlage (4) wird abgeleitet, dass Anträge stets von "DS" genehmigt werden. Dann wird jedoch durch Analyse von Vorlage (5) auch abgeleitet, dass Anträge immer von Akteuren der Gruppe "Administration" genehmigt werden. Letztere Regel ist offensichtlich redundant, da diese bereits in der *stärkeren* bzw. *spezielleren* direkten Zuweisungsregeln enthalten ist.

  Bestimmte Regeln sind in anderen Regeln enthalten

- Durch Analyse der Vorlagen (1) und (6) wird abgeleitet, dass ein Antrag im Allgemeinen gestellt werden muss bevor er genehmigt werden kann (*"S vor G"*). Doktoranden dürfen eine Unterkunft erst buchen, wenn der Antrag gestellt (*"S vor B"*) und genehmigt wurde (*"G vor B"*). Die Regel (*"S vor B"*) ist dabei redundant. Sie lässt bereits aus den beiden anderen Regeln automatisch folgern, d.h. *transitiv* ableiten.

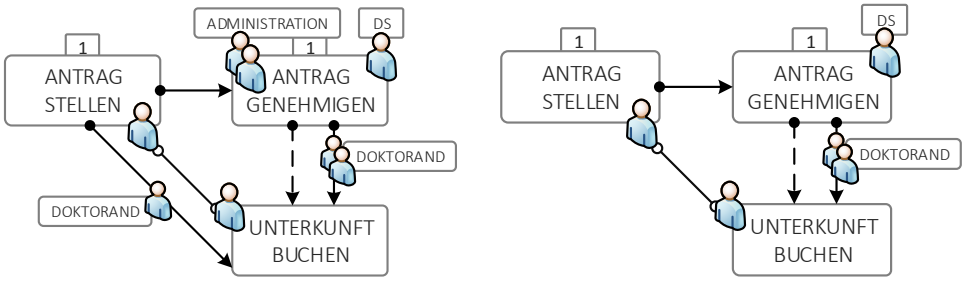
  Transitiv ableitbare Regeln

Identifikation redundanter Regeln

In einer Modell-Nachbearbeitungsphase werden derartige redundante Regeln identifiziert und aus dem Modell entfernt. Das Ergebnis ist ein einfacher verständliches Modell, dessen Aussage dabei unverändert bleibt. Die Bereinigung bzw. "Ausdünnung" (*engl. pruning*) des Beispielmodells ist in Abbildung 7b visualisiert.

1.6 ABGRENZUNG ZU VERWANDTEN ARBEITEN

Es erfolgt nun eine kurze Abgrenzung zu bestehenden Arbeiten. Eine detaillierte Betrachtung verwandter Arbeiten wird im weiteren Verlauf dieser Arbeit



(a) Abgeleitetes Modell **(b)** Nachbearbeitetes Modell

Abb. 7: Entfernen redundanter Regeln

gegeben und zusammenfassend in Kapitel 8 festgehalten. Ein Großteil existierender Process Mining-Methoden ist eher für den **Einsatz im Kontext von strikten Prozessen** geeignet [27], [80]. Wir bezeichnen diese Methoden als **klassische** Verfahren. Klassische Process Mining-Techniken wie der α-Algorithmus [7], der HeuristicsMiner [137] oder das genetische Process Mining [86] erzeugen Prozessmodelle in prozeduralen Modellierungssprachen wie Petri-Netzen. Der prozedurale Modellierungsansatz ist für agile Prozesse aber eher ungeeignet [27], [80]. Die Anwendung dieser Methoden auf die Ereignisprotokolle von agilen Prozessen liefert in vielen Fällen unüberschaubare Modelle (*"Spaghetti-Modelle"*), die durch Abstraktionsverfahren, wie der Filterung von Pfaden [58], nur ansatzweise verwendbar sind.

Prozedurale Verfahren

Der vorgestellte Ansatz kann zwei Forschungsrichtungen zuordnet werden: Regelbasiertem Process Mining und Process Mining mit Fokus auf die organisatorische Perspektive von Prozessen. Regelbasierte Process Mining-Verfahren sind besser für den Einsatz bei agilen Prozessen geeignet. In den letzten Jahren wurden einige Techniken zur automatisierten Ableitung von regelbasierten Prozessmodellen aus Ereignisprotokollen vorgeschlagen. Der *DeclareMiner* [81] extrahiert regelbasierte Modelle in der Sprache *Declare*. Es existieren Erweiterungen, die dessen Laufzeit beschleunigen [27], [79] und die Lesbarkeit sowie Ausdrucksstärke extrahierter Modelle verbessern [78]. Außerdem wurden weitere, sehr effiziente Algorithmen zur Ableitung von Declare-Modellen vorgestellt [45], [139]. Aufgrund der verwendeten Zielsprache Declare liegt der Fokus all dieser Arbeiten jedoch lediglich auf der Analyse der verhaltensorientierten Perspektive mit einigen Erweiterungen für Daten [80]. Die organisatorische Perspektive und deren tiefgreifender Einfluss auf den Prozessablauf werden in all diesen Arbeiten nicht betrachtet.

Regelbasierte Mining-Verfahren

Declare als Zielsprache

Keine Analyse der organisatorischen Perspektive

Komplementär zu diesen Arbeiten existieren Ansätze für die Analyse der organisatorischen Perspektive [144] in Prozessen. Diese verwenden die häufig in Protokollen vorhandene Information über Akteure, die bestimmte Aktivitäten durchführen. Mining-Methoden, die Ereignisprotokolle hinsichtlich organisatorischer Zusammenhänge analysieren, beschäftigen sich hauptsächlich mit der Erweiterung von prozeduralen Modellen durch organisatorische Information [125]. Außerdem existieren Verfahren zur Ableitung von einfachen Organisati-

Organisatorische Miner

onsmodellen [125] oder sozialen Netzwerken [5]. Auch Verfahren zur Analyse des Einflusses von Personen auf die Laufzeit bzw. die Performance von Prozessen existieren [93]. Ansätze, welche diese organisatorische Perspektive von Prozessen analysieren, sind bislang nicht in regelbasierte Mining-Verfahren integriert worden und sind daher für agile Prozesse ungeeignet.

Vereinigung beider Strömungen

Die Forschung dieser Dissertation vereinigt diese beiden Strömungen und beschreibt ein Process Mining-Verfahren zur Ableitung von regelbasierten, perspektivenübergreifenden Prozessmodellen. Die abgeleiteten Modelle beschreiben nicht nur komplexe organisatorische Zuweisungsregeln sondern geben außerdem Einblick in den tiefgreifenden Einfluss von Personen auf die Ausführungsreihenfolge von Aktivitäten.

1.7 ZUSAMMENFASSENDER ÜBERBLICK

Zusammenfassender Überblick

Abbildung 8 zeigt einen zusammenfassenden **Überblick** über den in dieser Dissertation entwickelten Process Mining-Ansatz für agile, personenbezogene Prozesse.

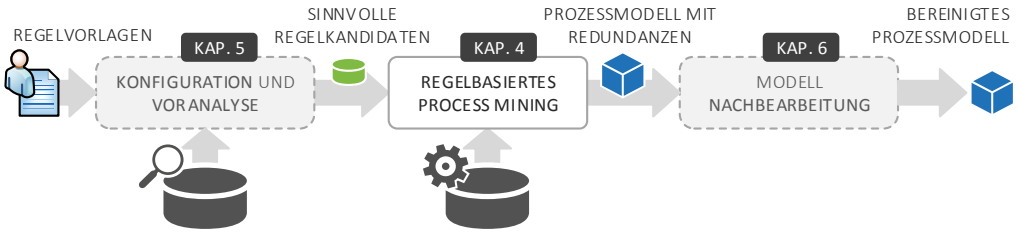


Abb. 8: Process Mining-Ansatz für agile, personenbezogene Prozesse

Erfüllung der Anforderungen

Auf Basis einer adäquaten Modellierungssprache werden Regel- bzw Suchvorlagen definiert. Diese stellen Schablonen für Zusammenhänge dar, nach denen in betrachteten Ereignisprotokollen gesucht werden soll. Wie in Abbildung 5 dargestellt, können auf diese Weise Vorlagen sowohl für komplexe organisatorische Zuweisungsregeln (Abbildung 5b) als auch für perspektivenübergreifende Zusammenhänge (Abbildung 5c) formuliert werden (A3 und A4). Sie enthalten Platzhalter für konkrete Entitäten, z.B. Aktivitäten, Akteure oder organisatorische Gruppen. In einer Voranalyse werden tatsächlich auftretende Kombinationen von Entitäten im betrachteten Log aufgesucht. Nur diese Kombinationen werden in die freien Platzhalter eingesetzt, wodurch unnötige Berechnung vermieden werden (A5). Das Einsetzen konkreter Werte in Platzhalter führt zur Erzeugung von Regelkandidaten. Im eigentlichen Mining-Vorgang wird anhand des Logs überprüft, welche Kandidaten nahezu immer erfüllt, tendenziell erfüllt und welche häufig verletzt sind. Erfüllte und lediglich tendenziell erfüllte Regeln werden in unterschiedlicher Modalität in das resultierende regelbasierte Prozessmodell übernommen (A1). Tendenziell erfüllte Regeln stellen keine verpflichtenden jedoch empfohlene Aktionen dar (A2). Zur Verbesserung der

Verständlichkeit werden in einer Nachbearbeitungsphase potentiell vorhandene, redundante Regeln entfernt (A5).

1.8 AUFBAU DER ARBEIT

Weiterer Aufbau der Dissertation

Der weitere Aufbau dieser Dissertation ist wie folgt: In Kapitel 2 werden die Grundlagen der automatisierten Datenanalyse im Allgemeinen und im Kontext des Prozessmanagements im Speziellen eingeführt. In Kapitel 3 werden verschiedene Modellierungsansätze und Sprachen evaluiert und eine adäquate Sprache zur Abbildung agiler, personenbezogener Prozesse ausgewählt. Auf Basis der gewählten Sprache werden in Kapitel 4 sowohl Konzept als auch Implementierung der Erzeugung und Überprüfung von Regelkandidaten erläutert. Außerdem werden Regelvorlagen verschiedener Perspektiven zur Ableitung häufig wiederkehrender Muster in Prozessen definiert. Kapitel 5 beschreibt, wie das Verfahren auf Basis unterschiedlicher Anforderungen konfiguriert und durch eine Voranalyse des gegebenen Logs die Komplexität vermindert werden kann. In Kapitel 6 wird beschrieben, wie abgeleitete Modelle nachträglich bearbeitet werden können, so dass redundante Regeln entfernt werden. In Kapitel 7 wird der vollständige Ansatz anhand verschiedener Ereignisprotokolle agiler Prozesse evaluiert. Kapitel 8 diskutiert detailliert verwandte Arbeiten und in Kapitel 9 werden die Ergebnisse dieser Dissertation schließlich zusammengefasst und ein Ausblick auf zukünftige Forschungsfragen gegeben.

2 GRUNDLAGEN DES PROCESS MINING

INHALT

In diesem Kapitel werden die Grundlagen der automatisierten Datenanalyse und Modellierung im Allgemeinen und im Kontext des Prozessmanagements im Speziellen eingeführt. Des Weiteren betrachten wir die Datenbasis für IT-gestützte Prozess-Analyseverfahren. Anschließend werden die Probleme und Schwächen klassischer Process Mining-Verfahren aufgezeigt.

2.1 MUSTERERKENNUNG DURCH DATA MINING

Analyse und strukturierte Darstellung digitaler Daten

Organisationen analysieren große Datenbestände um daraus wertvolle Informationen zu extrahieren, um Wettbewerbsvorteile zu erlangen, Effizienz zu steigern und um bessere Dienste anzubieten [50]. Durch die Analyse von großen Datenmengen erlangtes Wissen unterstützt Experten bei strategischen und operativen Entscheidungen. Um das vorhandene Wissen aus diesen Daten nutzbar zu machen, muss die darin enthaltene Information strukturiert und vereinfacht dargestellt werden.

2.1.1 Modelle als Abstraktion großer Datenmengen

Darstellung von Information durch Modelle

Um das Wissen, das in Datenmengen vorhanden ist, strukturiert und verständlich darzustellen, verwenden Experten häufig eine bestimmte Art von **Modell** [140]. Ein adäquates Modell bildet das Wissen und die Informationen der gegebenen Daten auf einfache und verständliche Weise in einer bestimmten Notation ab. Als **Modellierung** bezeichnet man daher den Prozess zur Erzeugung eines Modells aus einer gegebenen Datenmenge. Als Ergebnis der Modellierung wird die vereinfachte, pragmatische Abbildung eines Realitätsausschnittes gesehen [87]. Modelle ermöglichen es, neue und bislang unbekannte Einsichten in Zusammenhänge und ein besseres Verständnis des betrachteten Sachverhalts zu erlangen. Eine einfache Modellbildung ist beispielsweise das Auftragen von zweidimensionalen Datenpunkten in einem Graphen. Allein ein Blick auf die Datenwerte gibt mit hoher Wahrscheinlichkeit kaum Einblick in den zu Grunde liegenden funktionalen Zusammenhang. Durch die graphische Darstellung der Datenpunkte, d.h. durch die Bildung eines Modells, wird der funktionale Zusammenhang, beispielsweise in Form einer Parabel, hingegen offensichtlich.

Traditioneller, manueller Modellierungsprozess

Wenden wir uns uns nun dem Modellierungsprozess zu, d.h. der Frage wie Modelle gebildet werden. Die traditionelle Methode, um Wissen und aussagekräftige Informationen aus Daten zu extrahieren, basiert auf manueller Analyse, Interviews mit beteiligten Personen und Interpretationen [50]. Sei es im Bereich der Wissenschaft, des Marketings, des Finanzsektors, des Gesundheitswesens oder jeglichem anderen Bereich - der klassische Ansatz zur Analyse von Daten basiert stets darauf, dass einer oder mehrere Analysten Zugang und Verständnis zu Daten besitzen und als Schnittstelle zwischen Daten und Benutzern fungieren [50]. In vielen Fällen ist diese Art von **manueller Datenanalyse** sehr langsam, kostspielig und vor allem in höchstem Maße **subjektiv**. Mit zunehmendem Datenvolumen wird die manuelle Analyse von Daten mehr und mehr unpraktisch [50], [140].

Subjektive, manuelle Datenanalyse

2.1.2 Data Mining und der KDD-Prozess

Enorme Mengen digitaler Daten

Wir leben in einer Zeit in der Daten in enormen Mengen digital aufgezeichnet und gespeichert werden. Organisationen und Unternehmen speichern Daten über Unternehmensaktivitäten, Kunden, Mitarbeiter, Produkte, Produktionsabläufe, Geschäftspartner und viele weitere Bereiche digital in **Datenbanken** verschiedener Art ab (Relationale Datenbanken, strukturierte und unstrukturierte Dateien, etc.). Weil Computer Menschen befähigt haben derart viele Daten zu produzieren und aufzuzeichnen - mehr als wir manuell verarbeiten können, ist es nur natürlich, Computer auch dazu zu verwenden, **aussagekräftige** Muster und Strukturen aus diesen Daten zu extrahieren und diese zu Modellen zu vereinen. Ein Muster kann dabei als eine einzelne Komponente eines Modells angesehen werden, beispielsweise eine spezielle Regel (Muster) in einer Menge an Regeln (Modell) [140].

Automatisierte Analyse durch Data Mining

Der Begriff **Data Mining** bezeichnet Verfahren und Algorithmen, um die

Analyse einer großen Menge von Daten zu automatisieren, d.h. durch IT-Systeme zu unterstützen [50]. Diese Methoden analysieren den vorhandenen Datenbestand und suchen nach aussagekräftigen Beziehungen und Mustern in den Datensätzen. Abgeleitete Muster stellen eine verständliche Abstraktion der Daten dar und geben menschlichen Analysten Einblick in das enthaltene Wissen. Data Mining-Verfahren werden in allen denkbaren Bereichen von Wissenschaft, Forschung, Wirtschaft und Industrie angewendet. Ausgangspunkt sämtlicher Data Mining-Algorithmen sind *digital* vorhandene und zugängliche Datensätze. Im Gegensatz zu traditionellen Analysemethoden basieren die extrahierten Modelle sämtlicher Data Mining-Verfahren auf tatsächlich vorhandenen Daten, sind nahezu frei von subjektiver Interpretation und daher wesentlich **objektiver** als traditionalle Analysemethoden [50]. Der Nachteil dieser Verfahren besteht in der Abhängigkeit von Datenverfügbarkeit und Datenqualität.

Objektive Datenanalyse

Data Mining stellt eine vollständig datengetriebene Modellierungsmethode dar. Es ist offensichtlich, dass die Qualität der Resultate nicht nur von der Funktionalität der Analysealgorithmen sondern in großem Maße auch von der Qualität des Datenbestandes abhängt. Eine erfolgreiche Anwendung der automatisierten Modellierung mittels Data Mining verlangt daher viel mehr als nur die naive Anwendung von Algorithmen. Data Mining-Verfahren sind daher in einen Analyseprozess eingebettet, den sog. **Knowledge Discovery in Databases (KDD)**-Prozess [104]. Dieser Begriff wurde 1989 geprägt um herauszustellen, dass die Anwendung von Data Mining-Techniken nur ein Schritt in einem umfassenden Analyseprozess zur Extraktion von aussagekräftigen Informationen ist. KDD wurde dabei als der nicht-triviale Prozess der Entdeckung von **validen**, **neuen**, **potentiell nützlichen** und vor allem **verständlichen** Mustern in großen Datenbeständen definiert [104]. Abbildung 9 visualisiert die Schritte in diesem KDD-Prozess: Datenauswahl, Vorverarbeitung, Datentransformation sowie einer adäquaten Interpretation der Data Mining Ergebnisse [50].

Umfassender Datenanalyse-Prozess

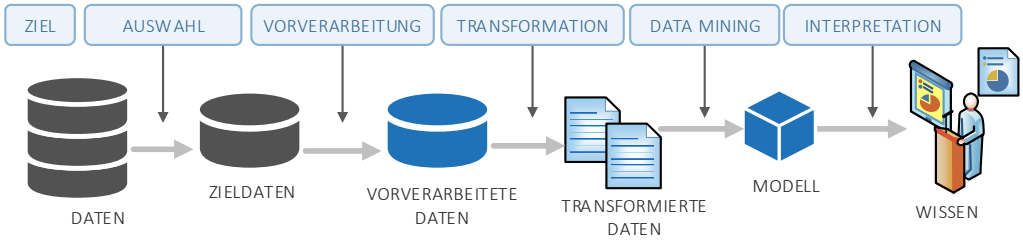


Abb. 9: KDD-Prozess zur Anwendung von Data Mining [50]

- Der erste Schritt besteht darin sich einen Überblick und ein grundlegendes Verständnis des Anwendungsbereichs zu verschaffen, vorhandenes **Vorwissen zu identifizieren** und das **Ziel** des Analyseprozesses zu definieren.
- Anschließend erfolgt die **Auswahl** der zu analysierenden Datenbestände, d.h. Selektion von Datensätzen und Datenattributen, die in die Analyse einbezogen werden sollen.

- Der dritte Schritt besteht in der **Vorverarbeitung** der ausgewählten Zieldaten. Hier wird versucht fehlerhafte Datensätze zu identifizieren und zu entfernen. Außerdem wird festgelegt, wie mit fehlenden Datenfeldern umzugehen ist.

- In der darauf folgenden Phase werden die Daten **transformiert**, d.h. in eine passende, dem Ziel der Analyse entsprechende Ausgangsform gebracht.

- Anhand der identifizierten Ziele aus Schritt 1 wird eine bestimmte **Data Mining**-Methode ausgewählt und auf den transformierten Datenbestand angewendet. Das Ergebnis sind Muster und Modelle in einer bestimmten Notation.

- Der letzte Schritt besteht darin die abgeleiteten Muster und Modelle zu visualisieren, zu interpretieren und darauf basierend zu **agieren**, d.h. das gewonnene Wissen zu nutzen.

Die zusätzlichen Schritte neben dem eigentlichen Data Mining-Schritt sind notwendig, um das Ableiten valider und aussagekräftiger Informationen bzw. Modelle aus den Datenbeständen sicherzustellen. Wird von diesem KDD-Prozess abgewichen, werden potentiell irrelevante Resultate geliefert (z.B. aufgrund fehlender Datenselektion) oder sogar fehlerhafte Ergebnisse abgeleitet (z.B. fehlender Datenvorverarbeitung bzw. falscher Transformation) [50].

2.2 PROCESS MINING

Data Mining ist ein Ansatz digital vorhandene Daten zu analysieren, um automatisiert Muster und Modelle zu extrahieren. Diese können Entscheidungsträger unterstützen und dabei helfen, die zunehmende Menge an digitalen Daten greifbar und überschaubar zu machen und einem Überfluss an Information entgegenwirken [140].

Process Mining

Der Begriff **Process Mining** bezeichnet die Anwendung von Data Mining-Methoden im Kontext von Prozessen [10]. Process Mining ist ein relativ neuer Forschungsbereich, angesiedelt zwischen Data Mining auf der einen Seite und der Modellierung und Analyse von Prozessen auf der anderen Seite und stellt daher eine Brücke zwischen diesen beiden Bereichen dar. Die Grundidee von Process Mining ist es, tatsächliche Prozesse, d.h. Arbeitsabläufe, durch automatisiertes Extrahieren von Modellen aus Ereignisprotokollen von IT-Systemen zu erkennen, zu überwachen und zu verbessern [11]. Das Wachstum der digitalen Welt, welches mit den Prozessen in Organisationen eng verbunden ist, ermöglicht es **(Prozess-)Ereignisse** (*engl. events*) digital aufzuzeichnen. Ereignisse beziehen sich auf eine Vielzahl von Vorgängen, wie beispielsweise das Abheben von Geld an einem Geldautomat, das Aktivieren eines Röntgengeräts durch einen Arzt, das Beantragen eines Führerscheins oder die Erstellung

Digitale Aufzeichnung von Prozessereignissen

eines elektronischen Flugtickets. Bei all den genannten Beispielen sind heutzutage IT-Systeme beteiligt und protokollieren eine Vielzahl der durchgeführten Aktivitäten. Die Herausforderung besteht darin, diese Ereignisdaten in einer geeigneten Form nutzbar zu machen, um Vorgänge besser zu verstehen, Engpässe zu identifizieren, Probleme und Abweichungen aufzuzeigen oder um Prozesse zu optimieren. Process Mining zielt genau auf diese Fragestellungen ab [11].

2.2.1 Typen von Process Mining-Verfahren

Process Mining nutzt die Daten, die während der IT-gestützten Durchführung von Prozessen protokolliert werden, zur Ableitung von Modellen [11]. Ausgangspunkt für Process Mining sind digitale, **ereignisorientierte Prozessausführungsprotokolle** (engl. Process Execution Logs), kurz **Logs**. Alle Methoden des Process Mining gehen davon aus, dass diese Art von Daten die zeitliche Reihenfolge der aufgezeichneten Ereignisse enthalten. Abbildung 10 gibt einen Überblick über den Einsatz von Process Mining und zeigt, dass Ereignisprotokolle auf drei verschiedene Arten genutzt werden können [11].

Prozessausführungsprotokolle (Logs)

- Die erste Art ist die **Extraktion** bzw. Entdeckung von Modellen (*engl. Process Discovery*). Eine Process Discovery-Methode verwendet Ereignisprotokolle als Eingabe, leitet daraus ein Modell ab und benötigt dazu keine weiteren Informationen. Das erzeugte Modell ist typischerweise ein Prozessmodell (z.B. ein BPMN-Modell), kann aber auch exklusiv andere Perspektiven beschreiben (z.B. ein soziales Netzwerk). Process Discovery ist der bekannteste Anwendungsbereich des Process Mining.

Modell-Entdeckung (Process Discovery)

- Die zweite Form von Process Mining ist die **Konformitätsüberprüfung** (*engl. Conformance Checking*). Hierbei wird ein bestehendes Modell des Prozesses mit einem gegebenen Ereignisprotokoll verglichen. Process Mining-Verfahren werden hier verwendet, um zu bestimmen, inwieweit die in den Logs protokollierten Prozessdurchführungen mit dem Modell übereinstimmen. Diese Methoden benötigen Ereignisprotokolle und ein Modell als Eingabe. Die Ausgabe umfasst diagnostische Informationen mit Blick auf Unterschiede und Übereinstimmungen zwischen Modell und Logs.

Konformitätsüberprüfung

- Der dritte Typ des Process Mining ist die **Verbesserung** (*engl. Enhancement*). Ein gegebenes Modell wird hier auf Basis der Informationen des Ereignisprotokolls verbessert oder erweitert. Ein Typ der Verbesserung ist die **Fehlerbehebung** (*engl. repair*), d.h. das Modell wird so modifiziert, dass es die Realität besser abbildet. Wird in einem Modell beispielsweise abgebildet, dass zwei Aktivitäten sequentiell durchgeführt werden müssen, diese jedoch in der Realität in beliebiger Reihenfolge abgearbeitet werden, so kann das Modell in dieser Hinsicht angepasst werden. Der zweite Typ der Modell-Verbesserung ist die (**Modell-Erweiterung** (*engl. extension*). Hier wird das Modell beispielsweise durch eine weitere Per-

Modell-Verbesserung

spektive erweitert. Enthält ein bestehendes Modell lediglich Informationen zur Ausführungsreihenfolge der Aktivitäten, d.h. der verhaltensorientierten Perspektive, so kann das Modell durch Analyse der organisatorischen Perspektive durch Ressourcen-Zuweisungsregeln erweitert werden. Diese Methoden benötigen ebenso ein Modell und ein Log als Eingabe. Die Ausgabe ist ein erweitertes bzw. repariertes Modell.

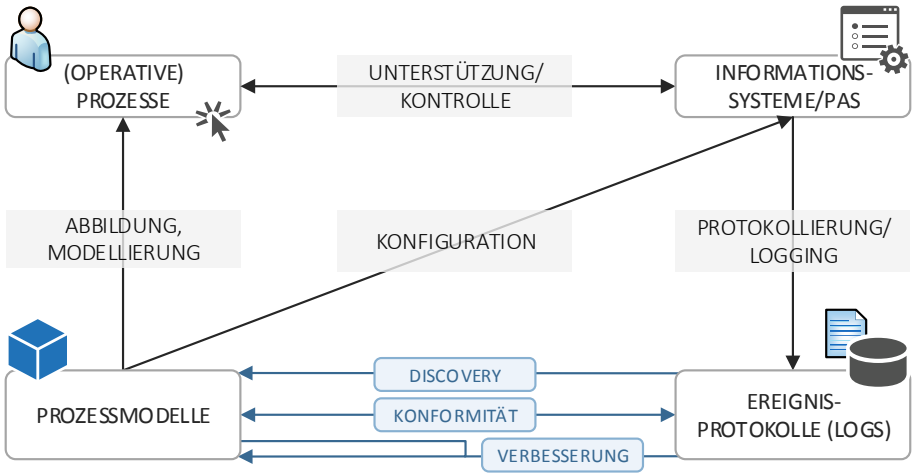


Abb. 10: Verschiedene Arten des Process Mining [10]

2.2.2 Geschichte des Process Mining

Wurzeln in diversen Forschungsrichtungen

Process Mining ist aus verschiedenen Forschungsrichtungen entstanden. Einerseits hat es seine Wurzeln in den Bereichen der logischen Inferenz und der Stochastik, andererseits in den Bereichen des Data Mining und des maschinellen Lernens. Bereits 1967 beschreibt Mark Gold in [54] mehrere logische Inferenz Probleme. Die Herausforderung lag dabei darin, Regeln auf Basis einer infiniten Kette an Beispielen zu folgern.

Apriori-Algorithmus

Mitte der 90-Jahre entwickelten Forscher wie Rakesh Agrawal mehrere Data Mining-Algorithmen, um häufige Muster in großen Datenbeständen zu finden. In [14] wurde der bekannte Apriori-Algorithmus zur Ableitung von Assoziationsregeln vorgestellt. Dieser wurde in [15] zur Entdeckung von häufigen Sequenzen und häufigen Episoden [84] erweitert. Keine dieser Methoden ist jedoch in der Lage ein vollständiges Modell eines Prozesses zu entdecken, da lediglich lokale Muster analysiert werden. Außerdem wird lediglich der Kontrollfluss des Prozesses betrachtet.

Entdeckung von Softwareprozessen

Cook und Wolf entwickelten in der zweiten Hälfte der 90-Jahre die ersten Techniken zur Entdeckung von vollständigen Prozessmodellen im Kontext von Software-Entwicklungsprozessen. In [39] beschreiben sie dazu drei Methoden zur Ableitung von Prozessmodellen: Eine unter Verwendung von neuronalen Netzen, eine welche rein algorithmisch vorgeht und eine auf Basis eines Markov-Ansatzes. Alle diese Ansätze sind auf rein sequentielle Zusammenhänge der

verhaltensorientieren Perspektive beschränkt, d.h. es kann keine Nebenläufigkeit bzw. Parallelität entdeckt werden.

Enteckung von Geschäftsprozessmodellen

Im Jahr 1998 erschienen erstmalig zwei Veröffentlichungen [13], [41], welche unabhängig voneinander Techniken zur Entdeckung von Prozessen im Kontext des Geschäftsprozessmanagements vorschlugen. Der Ansatz in [13] entdeckt kausale Abhängigkeiten zwischen Aktivitäten, kann jedoch keine parallelen oder exklusiven Verzweigungen oder Zusammenführungen ableiten. Auch der Ansatz in [41] ist lediglich in der Lage sequentielle Prozessmodelle zu entdecken. Herbst war einer der ersten Forscher, der einen Ansatz zur Ableitung komplexerer prozeduraler Prozessmodelle vorstellte [62]. Er schlug vor, stochastische Graphen als Übergangsrepräsentation zu verwenden, um daraus anschließend ein Prozessmodell in der ADONIS-Notation zu generieren. Die Graph-Erzeugung funktioniert dabei ähnlich wie in [13]. Die meisten der klassischen Ansätze haben Probleme Nebenläufigkeit, d.h. parallel ablaufende Aktivitäten, zu erkennen.

Process Mining-Verfahren

Der α-Algorithmus [7] und der HeuristicMiner [137] der Eindhovener Forschungsgruppe um Wil van der Aalst waren die ersten Verfahren die, nun unter dem Begriff *Process Mining*, Nebenläufigkeit betrachteten. Der Fokus des HeuristicMiners liegt dabei darauf, auch mit teilweise fehlerhaften (*engl. noise*) oder unvollständigen (*engl. incompleteness*) Ereignisprotokollen umgehen zu können [137]. Um die praktische Anwendbarkeit von prozeduralen Process Mining-Verfahren zu gewährleisten, ist es wichtig dies zu berücksichtigen. Weitere prozedurale Verfahren, die auf Basis von Heuristiken mit unvollständigen und (evtl.) fehlerhaften Daten arbeiten können, sind neben dem HeuristicMiner auch der FuzzyMining-Ansatz [58] und das genetische Process Mining [86]. All diese Verfahren erzeugen prozedurale Prozessmodelle, welche sich vor allem für die Analyse von relativ strikten Prozessen eignen.

2.3 DATENGRUNDLAGE FÜR PROCESS MINING-VERFAHREN

Ereignisprotokolle als Grundlage

Ereignisprotokolle stellen die Grundlage für Process Mining-Methoden dar. Process Mining ist nicht möglich ohne die Verfügbarkeit vernünftiger Protokolle. Abhängig von der verwendeten Process Mining-Technik bestehen unterschiedliche Anforderungen an die zu analysierenden Daten. Je nach Fragestellung, welche durch eine Process Mining-Anwendung beantwortet werden soll, werden außerdem unterschiedliche Sichten auf die verfügbaren Prozessdaten benötigt. In den folgenden Abschnitten widmen wir uns daher den Datenquellen für Process Mining-Verfahren.

2.3.1 Datenquellen

Verschiedenste Quellen

Process Mining-Methoden analysieren Ereignisdaten aus einer prozessorientierten Perspektive [10]. Ausgangspunkt für Process Mining sind digitale Daten, die aus verschiedenen Quellen bezogen werden können. Eine geeignete Daten-

quelle kann eine einfache Textdatei sein, eine Excel-Datei, ein Transaktionsprotokoll oder eine Tabelle in einer relationalen Datenbank. Daten können auch aus Webseiten, E-Mails, PDF-Dokumenten oder gescannten Textdokumenten extrahiert werden. Man sollte jedoch nicht erwarten, dass alle verwertbaren Daten in einer einzigen, wohlgeformten Datenbank abgelegt sind. In der Realität sind die Daten über verschiedene Datenquellen verteilt. Häufig ist ein großer Aufwand notwendig, um verwertbare Protokolldaten zu extrahieren [10]. Um Prozesse über Unternehmensgrenzen entlang einer gesamten Wertschöpfungskette hinweg analysieren zu können, müssen beispielsweise sogar Datenquellen mehrerer Organisationen verwendet und integriert werden. Aufgrund der in vielen Fällen enormen Menge an verfügbaren Daten, sollten bereits bei der Gewinnung von Daten klare Ziele und Fragestellungen vorliegen, die durch die Analyse beantwortet werden sollen. Anstatt alle verfügbaren Daten zu sammeln, werden Daten nur auf Basis der Ziele zusammengetragen.

Ereignisdaten sind häufig verteilt

Klare Ziele bei Gewinnung von Daten

ETL-Prozess

Im Bereich des Business Intelligence (BI) und des klassischen Data Mining wird die Extraktion brauchbarer Daten im ETL-Prozess (*Extract, Transform, Load*) beschrieben, der folgende Schritte beinhaltet: (i) Extrahieren von Daten aus externen Datenquellen (*Extraction*), (ii) Transformation der Daten in ein adäquates Format (*Transformation*), (iii) Laden der Daten in das Zielsystem, zum Beispiel in ein Data Warehouse oder eine Datenbank (*Load*). In vielen Fällen ist ein großer Aufwand notwendig um eine einheitliche Sicht auf die extrahierten Daten zu erzeugen. Verschiedene Datenquellen verwenden potentiell unterschiedliche Bezeichner oder Formatierungskonventionen. In einer Datenquelle wird beispielsweise das Datumsformat "31-12-2014"verwendet, wohingegen in einer anderen das Format "2014-12-31"verwendet wird. Diese Hindernisse sind, wie im klassischen Data Mining, auch im Bereich des Process Mining zu überwinden.

Großer Aufwand zur Erzeugung einheitlicher Sichten

2.3.2 Der Process Mining-Prozess

Abbildung 11 gibt einen Überblick über den Process Mining-Prozess und ordnet die einzelnen Phasen dem klassischen KDD-Prozess zu. Die verfügbaren Daten müssen *extrahiert* werden und zu analysierbaren Ereignisprotokollen konvertiert werden. Ein passendes Datenformat für Ereignisprotokolle ist beispielsweise das *XES*-Format, welches in Abschnitt 2.3.4 erläutert wird. Diese Extraktionsphase entspricht der Datenauswahl im KDD-Prozess. Wie bereits erwähnt ist dabei die Einschränkung auf eine interessante, dem Ziel der Analyse entsprechenden Menge an Daten von enormer Wichtigkeit. Wir beschränken uns daher darauf, dass ein Ereignisprotokoll zu einem bestimmten Prozess gehört. Während der Extraktionsphase sind daher nur die Ereignisdaten relevant, die dem zu analysierenden Prozess zuzuordnen sind (*Grobe Filterung der Daten*).

Extraktion von Daten

Abhängig von den Fragen, die durch die Analyse beantwortet werden sollen, können aus derselben Menge an Daten auch unterschiedliche Ereignisprotokolle erzeugt werden. Betrachtet man beispielsweise die Daten und Informationen, die in einer Klinik vorliegen. Analysten können hier an verschiedenen Zusam-

Verschiedene Ereignisprotokolle aus gleichen Daten

menhängen interessiert sein, z.B. an der Modellierung von Ablaufpfaden von Behandlungen. Für diese Art von Analyse werden lediglich Informationen zu Behandlungsschritten und entsprechende Zeitstempel benötigt, um die Reihenfolge des Behandlungsprozesses rekonstruieren zu können. In anderen Fällen will man jedoch zusätzlich die Rolle von behandelten Ärzten und Pflegepersonal in diesem Behandlungsprozess analysieren, um beispielsweise deren Auslastung oder bestimmte Abhängigkeiten zu Behandlungsschritten offenzulegen. In diesem Fall werden zusätzlich Informationen zum durchführenden Personal für jede Aktivität benötigt. Zur Beantwortung beider Fragestellungen werden offensichtlich unterschiedliche Ausprägungen von Ereignisprotokollen benötigt oder gegebene Protokolle auf unterschiedliche Weise analysiert [10].

Feine Filterung von Daten

Nachdem ein Ereignisprotokoll für den zu analysierenden Prozess erzeugt wurde, wird es typischerweise gefiltert. Im Groben werden die verfügbaren Daten bereits während der Extraktion aus den Quellsystemen gefiltert. Die *Filterung* von Daten bezieht sich auf eine *feinere Filterung* der bestehenden Daten auf Basis von ersten Analyseresultaten. Analysten können sich beispielsweise dazu entscheiden, lediglich die Ereignisse der am häufigsten im Ereignisprotokoll auftretenden Aktivitäten in die Analyse einzubeziehen. Auf diese Weise kann die Größe des abgeleiteten Modells eingeschränkt werden und somit die Überschaubarkeit durch Abstraktion von seltenem und daher eher irrelevantem Verhalten verbessert werden [58]. Die Filterung der Daten ist der Vorverarbeitungsphase des KDD-Prozesses zuzuordnen. Das gefilterte Ereignisprotokoll stellt schließlich die Eingangsdaten für verschiedene Process Mining-Techniken dar. Je nach Ziel der Analyse können unterschiedliche Verfahren mit unterschiedlicher Konfiguration gewählt werden. Das Ergebnis einer Process Mining-Analyse ist schließlich ein Prozessmodell, welches interpretiert, nachbearbeitet und ausgeführt werden kann.

Process Mining

Interpretation von erzeugten Modellen

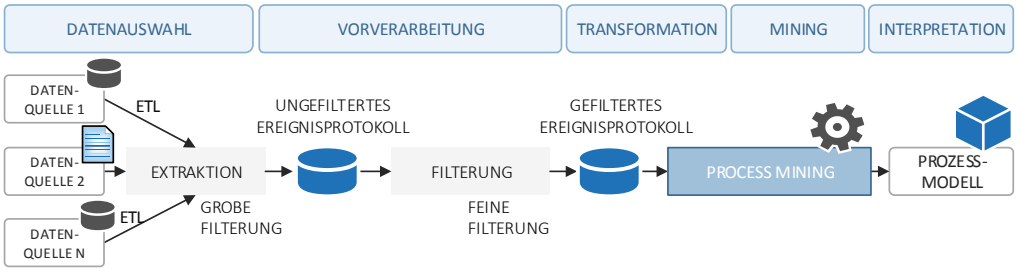


Abb. 11: Übersicht über den Process Mining-Prozess

2.3.3 Ereignisprotokolle

In Tabelle 2 ist ein Ausschnitt eines Ereignisprotokolls dargestellt. Diese Tabelle zeigt die Informationen, die typischerweise in einem Ereignisprotokoll enthalten sind, das für Process Mining-Verfahren verwendet werden kann [10]. Das Protokoll enthält exemplarische Ereignisse eines universitären Dienstreiseabwicklungsprozesses, welche als "Nebenprodukt"der IT-gestützten Durch-

führung des Prozesses anfallen. Allen Analysen der folgenden Kapitel legen wir zu Grunde, dass die Ereignisse des betrachteten Ereignisprotokolls stets einem bestimmten Prozess zuzuordnen sind. Die grobgranulare Filterung sollte daher sicherstellen, dass es sich bei den extrahieren Ereignisdaten um Ereignisse des zu analysierenden Prozesses handelt. Jedes Ereignis ist dabei einer bestimmten *Aktivität* zugeordnet. Des Weiteren wird aus der Tabelle ersichtlich, dass jedes Ereignis im Protokoll einer bestimmten *Prozessinstanz*, d.h. einem bestimmten *Fall* des Prozesses zugeordnet ist. In Tabelle 2 ist beispielsweise jede Antragsstellung (Aktivität) einer bestimmten Dienstreise (Instanz/Fall) zugeordnet. Eindeutige Bezeichner für die Prozessinstanz, d.h. den Fall (*engl. Case ID*), und die Aktivität (*engl. Activity ID*) stellen eine grundlegende Anforderung an Ereignisprotokolle dar, damit diese durch Process Mining-Verfahren analysiert werden können. Des Weiteren müssen die Ereignisse innerhalb einer Prozessinstanz *geordnet* sein. Ohne eine Ordnung der Ereignisse können keine kausalen Abhängigkeiten zwischen Aktivitäten gefolgert werden. In Tabelle 2 sind außerdem weitere Informationen zu jedem Ereignis gespeichert. Beispielsweise haben alle Ereignisse einen *Zeitstempel* und einen bestimmten *Ereignistyp*, der die Art des Ereignisses beschreibt (z.B. *Start* oder *Complete* einer Aktivität). Diese Information kann nützlich sein, um beispielsweise die durchschnittliche Durchlaufzeit des Prozesses oder die durchschnittliche Dauer einer Aktivität zu berechnen. Außerdem enthält das Protokoll typischerweise Informationen zu *Ressourcen* (hier abgekürzt durch "Res."), d.h. den ausführenden bzw. verantwortlichen Personen einer Aktivität.

Ereignisse genau einem Prozess zugeordnet

Prozessinstanz

Weitere protokollierte Informationen

Zusammenfassend lässt sich der zu Grunde gelegte Inhalt eines Ereignisprotokolls für Process Mining wie folgt beschreiben:

- Ein Prozess besteht aus mehreren Prozessinstanzen (*Fällen*).
- Eine Prozessinstanz besteht aus mehreren Ereignissen von Aktivitäten, so dass sich jedes Ereignis genau einer Instanz zuordnen lässt.
- Die Ereignisse innerhalb einer Instanz sind geordnet.
- Für Ereignisse sind zusätzliche Informationen (*Attribute*) vorhanden, wie z.B. ID der Aktivität, Zeitstempel und durchführende bzw. verantwortliche Person.

Formailisierung der Begriffe

Im folgenden Abschnitt wollen wir die verschiedenen Begriffe, welche im Kontext von Ereignisprotokollen auftreten, formalisieren.

- Ein *Ereignis* $e = \{\mathrm{id}, \mathrm{case}, \mathrm{type}, \mathrm{activity}, \mathrm{resource}, \mathrm{time}\}$ ist ein Tupel, das durch einen eindeutigen Bezeichner *id* gekennzeichnet ist, einer Prozessinstanz *case* zugeordnet ist, einen speziellen Ereignistyp *type* besitzt, einer bestimmten Aktivität *activity* zuzuordnen ist und von einer bestimmten Ressource *resource* zu einer bestimmten Zeit *time* durchgeführt wurde.
- Eine *Spur* $\sigma = \langle e_1, e_2, ..., e_n \rangle$ bezeichnet eine geordnete Menge an Ereignissen, die während der Ausführung einer einzigen Prozessinstanz erzeugt wurde. $|\sigma_i|$ bezeichnet die Länge einer Spur σ_i.

Tabelle 2: Auszug eines Ereignisprotokolls eines Dienstreiseabwicklungsprozesses

Fall	Zeitstempel	Ereignistyp	Aktivität	Res.	...
SS-Riga2013	2013-07-06T15:12	Start	Antrag stellen	SS	...
	2013-07-06T15:31	Complete	Antrag stellen	SS	...
	2013-07-07T16:22	Start	Antrag prüfen	KH	...
	2013-07-07T16:28	Complete	Antrag prüfen	KH	...
	...	...	...	...	...
	2013-07-09T10:22	Start	Antrag genehmigen	DS	...
	2013-07-09T10:28	Complete	Antrag genehmigen	DS	...
	2013-07-10T09:02	Start	Unterkunft buchen	SS	...
	2013-07-10T09:45	Complete	Unterkunft buchen	SS	...
	2013-07-11T10:24	Start	Antrag archivieren	KH	...
	2013-07-11T10:29	Complete	Antrag archivieren	KH	...
SJ-Muc2014	2014-11-06T14:58	Start	Unterkunft buchen	SJ	...
	2014-11-06T15:06	Complete	Unterkunft buchen	SJ	...
	2014-11-08T15:36	Start	Antrag stellen	SJ	...
	2014-11-08T15:59	Complete	Antrag stellen	SJ	...
	2013-07-09T09:16	Start	Antrag prüfen	KH	...
	2013-07-09T09:22	Complete	Antrag prüfen	KH	...
	...	...	...	...	...
	2014-11-16T10:11	Start	Antrag genehmigen	DS	...
	2014-11-16T10:24	Complete	Antrag genehmigen	DS	...
	2014-11-17T12:33	Start	Antrag archivieren	KH	...
	2014-11-17T12:59	Complete	Antrag archivieren	KH	...
...	...	...	...	...	...

- Ein *Ereignisprotokoll* $\Phi = \{\sigma_1, \sigma_2, ...\sigma_k\}$ besteht schließlich aus einer Menge an Spuren und beschreibt exakt die Ausführung aller Prozessinstanzen innerhalb eines bestimmten Zeitraums.

Ereignisinformation zu Daten und Applikationen

Da in dieser Arbeit die datenorientierte und die operationale Perspektive nicht betrachtet werden, sind Ereignisinformationen zu diesen Perspektiven nicht Bestandteil betrachteter Ereignisprotokolle. Zur übersichtlichen Darstellung von Ereignisprotokollen verwenden wir in dieser Arbeit eine abkürzende Notation. Die aufgezeichneten *Start*- und *Complete*-Ereignisse einer bestimmten Aktivität a, die von einer Person p durchgeführt wurden, werden als **s**(a,p) bzw. **c**(a,p) bezeichnet. Des Weiteren sind die aufgeführten Ereignisse bereits temporär geordnet, so dass Zeitstempel nicht mehr explizit angegeben werden. Die in Tabelle 2 beschriebene Spur σ_1="*SS-Riga2013*"wird daher wie folgt abgebildet:

Abkürzende Notation für Protokolle

σ_1: ⟨**s**(Antrag stellen,SS), **c**(Antrag stellen,SS), **s**(Antrag prüfen,KH), **c**(Antrag prüfen,KH),..., **s**(Antrag genehmigen,DS), **c**(Antrag genehmigen,DS), **s**(Unterkunft buchen,SS), **c**(Unterkunft buchen,SS), **s**(Antrag archivieren,KH), **c**(Antrag archivieren,KH)⟩.

2.3.4 eXtensible Event Stream (XES)

Das Standardformat zur Speicherung und zum Austausch von Ereignisprotokollen ist seit 2010 das XML-basierte ***eXtensible Event Stream***, kurz *XES* [59], [136]. In Abbildung 12 ist das Meta-Modell von XES in einem UML-Klassendiagramm dargestellt. Ein XES-Dokument, d.h. eine XML-Datei, stellt genau ein Ereignisprotokoll eines Prozesses dar, dass aus mehreren Spuren besteht. Jede Spur beschreibt eine sequentielle Liste an Ereignissen, die einer bestimmten Prozessinstanz zuzuordnen sind. Ereignisprotokoll, Spuren und Ereignisse können Attribute besitzen, welche jeweils einen bestimmten Datentyp haben, namentlich String, Date, Int, Float und Boolean. XES gibt im Allgemeinen keine feste Menge an verpflichtenden Attributen vor. Um jedoch bestimmte Attribute semantisch zuordnen zu können, werden Erweiterungen (*engl. extensions*) definiert. Standardmäßig enthaltene Erweiterungen sind die *Time*-Erweiterung, welche beispielsweise ein Zeitstempel-Attribut definiert und die *Organizational*-Erweiterung, die ein Attribut für die durchführende Ressource definiert. Benutzer können jedoch auch eigene, domänenspezifische Erweiterungen definieren. In XES können bestimmte Attribute als verpflichtend deklariert werden. Beispielsweise kann angegeben werden, dass jede Spur einen eindeutigen Namen und jedes Ereignis einen Zeitstempel besitzen muss. Zu diesem Zweck werden in einer XES-Datei zwei verschiedene Listen an *globalen Attributen* definiert. Eine Liste enthält die verpflichtenden Attribute für Spuren, die andere Liste die verpflichtenden Attribute für Ereignisse.

Meta-Modell von XES

Attribute

Erweiterungen (extentions)

Globale Attribute

Das XES-Meta-Modell in Abbildung 12 gibt keine konkrete Syntax vor. Prinzipiell sind verschiedene Serialisierungsvarianten möglich, eine XML-basierte Serialisierung hat sich jedoch etabliert. Listing 1 zeigt einen Ausschnitt der XES XML-Serialisierung des Ereignisprotokolls aus Tabelle 2. In diesem Beispiel sind vier Erweiterungen definiert: *Concept*, *Lifecycle*, *Organizational* und *Time*. Für jede Erweiterung wird dabei ein Präfix angegeben.

XML-basierte Serialisierung von XES

- *Concept-Erweiterung* Die *Concept*-Erweiterung (Präfix *concept*) definiert das *name*-Attribut für Spuren und Ereignisse. Im Fall von Spuren wird dadurch typischerweise ein eindeutiger Bezeichner für die jeweilige Prozessinstanz (z.B. die ID der Dienstreise) angegeben. Im Fall von Ereignissen wird dadurch typischerweise der Name der zugehörigen Aktivität (z.B. Antrag stellen) angegeben.
- *Lifecycle-Erweiterung* Die *Lifecycle*-Erweiterung (Präfix *lifecycle*) definiert das *transition*-Attribut für Ereignisse. Durch dieses Attribut wird typischerweise der Typ des Ereignisses, z.B. *Start* oder *Complete*, angegeben.
- *Organizational-Erweiterung* Die *Organizational*-Erweiterung (Präfix *org*) definiert beispielsweise das *resource*-Attribut für Ereignisse, durch das die durchführende bzw. verantwortliche Person des jeweiligen Ereignisses angegeben wird.
- *Time-Erweiterung* Die *Time*-Erweiterung (Präfix *time*) definiert das *timestamp*-Attribut für Ereignisse, wodurch Datum und Zeit eines Ereignisses angegeben werden.

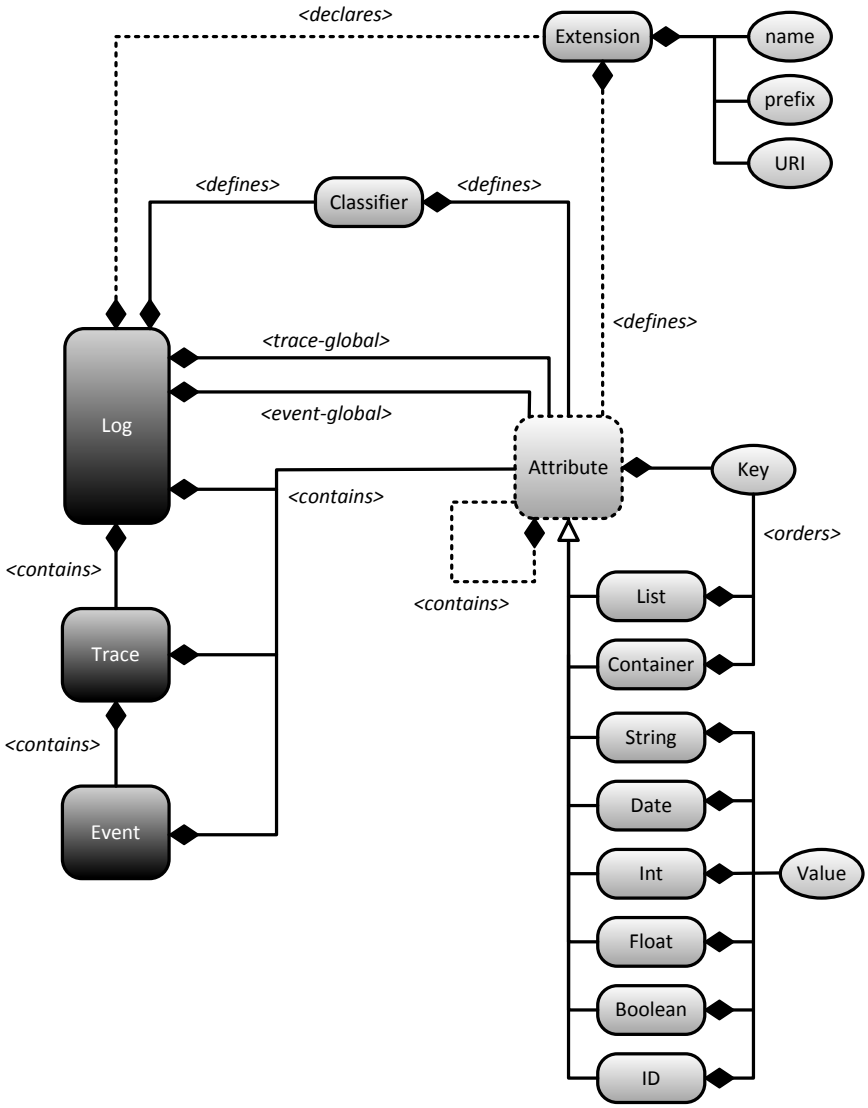


Abb. 12: Meta-Modell des XES-Formats [59]

Des Weiteren werden in der XES-Datei globale, verpflichtende Attribute definiert. Das Attribut *concept:name* ist für alle Spuren verpflichtend, wohingegen die Attribute *concept:name*, *org:resource* und *time:timestamp* für alle Ereignisse angegeben sein muss. Für weitere Informationen zur XES-Syntax verweisen wir auf [59].

Listing 1: Auszug eines XML-basierten XES-Ereignisprotokolls

```
<log>
<extension name="Concept" prefix="concept" uri="..." />
<extension name="Time" prefix="time" uri="..." />
<extension name="Organizational" prefix="org" uri="..." />
<global scope="trace">
        <string key="concept:name" value="name" />
</global>
```

```
<global scope="event">
        <string key="concept:name" value="name" />
        <string key="org:resource" value="resource" />
        <string key="time:timestamp" value="2013-06-06T15:31:00.000+01:00" />
</global>
<trace>
<string key="concept:name" value="SS_Riga2013"/>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-07-06T15:12:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-07-06T15:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-07-07T16:22:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-07-07T16:28:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
...
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-07-09T10:22:00.000+01:00"/>
        <string key="concept:name" value="Antrag genehmigen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-07-09T10:28:00.000+01:00"/>
        <string key="concept:name" value="Antrag genehmigen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-07-10T09:02:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
```

```
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-07-10T09:45:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-07-11T10:24:00.000+01:00"/>
        <string key="concept:name" value="Antrag archivieren"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-07-11T10:29:00.000+01:00"/>
        <string key="concept:name" value="Antrag archivieren"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
...
</trace>
...
</log>
```

2.3.5 Akquisition von Ereignisprotokollen

In den vorherigen Abschnitten wird gezeigt, dass ein digital zugängliches, konsistentes und möglichst vollständiges Ereignisprotokoll historischer Prozessabläufe die Grundlage für Process Mining-Verfahren darstellt. Die Qualität von abgeleiteten Modellen ist in großem Maße abhängig von der Qualität der Ereignisprotokolle. Vor allem in von menschlichen Handlungen geprägten Prozessen sind Ereignisprotokolle häufig nicht vollständig, nicht zugänglich oder nicht digital vorhanden [10], [127]. Hier werden Prozesse nur teilweise von Informationssystemen unterstützt, wodurch nur Teile des Prozesses digital aufgezeichnet und für Process Mining verwendet werden können [9]. Viele Prozesse sind wenig greifbar und nur implizit durch Handlungen von Akteuren gegeben [9]. Ein „digitaler Fußabdruck", den Process Mining-Verfahren benötigen, ist in vielen Fällen nicht vorhanden. Die Problematik der Bereitstellung hochwertiger Ereignisprotokolle befindet sich unter den Schlüsselherausforderungen der Task Force on Process Mining [10].

Qualität der Ereignisprotokolle

Unzureichende Qualität bei personenbezogenen Prozessen

Diese Tatsache lässt sich am Beispiel von medizinischen Prozessen belegen. Ein Arzt plant die Behandlung eines Patienten in seinem eigenen Ermessen selbst und weitestgehend unabhängig. Prozesse im Gesundheitswesen werden geprägt von menschlichen Entscheidungsträgern und sind daher weniger greifbar und weniger digital nachvollziehbar als maschinelle, strikte Abläufe [74]. Nur ein Teil der klinischen Behandlungsabläufe wird durch IT-Systeme unterstützt. Bei den zur Untersuchung verwendeten Systemen und Anwendungen handelt es sich häufig um autonome, d.h. unabhängig voneinander entwickelte und nicht integrierte Systeme [85], oder Altsysteme, welche keine extrahier-

baren Ereignisprotokolle erzeugen und aufwändig adaptiert werden müssen [100]. Ereignisprotokolle derartiger Prozesse sind daher in vielen Fällen nicht zugänglich, nicht vollständig oder nicht vorhanden.

Regelfreie Prozessausführung bzw. Prozess-Observation

Eine Methode zur Akquisition von qualitativ hochwertigen Ereignisprotokollen wird in Kapitel 3.3.5 erläutert und als *regelfreie Prozessausführung* bzw. *Prozess-Observation* (*engl. Process Observation*) bezeichnet.

Wunschlinien (engl. desire lines)

Eine passende Metapher für die Prozess-Observation sind Wunschlinien (*engl. desire lines*) bzw. "Trampelpfade" [2]. Trampelpfade werden von Menschen unabhängig von bestehenden, ausgebauten Wegen begangen, weil sie beispielsweise geeigneter, zweckmäßiger oder kürzer sind. Häufig sind diese Pfade anders als ausgebaute Wege. Breite und Abnutzung geben Anhaltspunkte darüber, wie häufig ein Trampelpfad benutzt wurde. Der Begriff "Wunschlinie" wurde bereits jahrzehntelang im Bereich der Stadtplanung verwendet. Planer nahmen vorhandene Trampelpfade als Grundlage, um befestigte Wege zu bauen. Die Wege durch den Central Park von New York entstanden beispielsweise auf diese Weise [91]. Durch Prozess-Observation werden derartige Wunschlinien in Prozessen sichtbar gemacht und anschließend potentiell zu festen Wegen, d.h. standardisierten Prozessmodellen ausgebaut bzw. erweitert werden.

2.4 KLASSISCHE PROCESS MINING-VERFAHREN

Process Discovery-Verfahren

Während des letzten Jahrzehnts sind zahlreiche Process Mining-Verfahren zur automatisierten Modellierung von Prozessen entwickelt und in unterschiedlichen Einsatzgebieten evaluiert worden [11]. Der Großteil der Verfahren strebt dabei die Extraktion eines Prozessmodells an (*engl. Process Discovery*). Nach van der Aalst [10] ist das generelle Ziel eines *Process Discovery*-Algorithmus, aus einem Ereignisprotokoll Φ, wie der XES-Datei in Listing 1, ein Prozessmodell zu extrahieren, welches das Verhalten des Protokolls abbildet. Diese Definition spezifiziert dabei nicht, welche Art von Prozessmodell erzeugt werden soll (Flussdiagramm, logische Regeln, etc.) und mit welcher Notation bzw. Sprache (Petri-Netz, Lineare temporale Logik, etc.) das resultierende Modell dargestellt werden soll. Des Weiteren wird nicht spezifiziert, welche Modellierungsperspektiven einbezogen werden sollen.

Klassische Process Mining-Techniken

Klassische Process Mining Techniken wie der α-Algorithmus [7], der HeuristicsMiner [137] oder das genetische Process Mining [86] erzeugen Prozessmodelle in prozeduralen Modellierungssprachen wie Petri-Netzen oder Abhängigkeitsgraphen (*engl. dependency graph*). Exemplarisch für prozedurale Verfahren wird im folgenden Abschnitt der **α-Algorithmus** [7] erläutert, welcher lediglich die **verhaltensorientierte Perspektive** eines zu analysierenden Prozesses betrachtet. Anschließend werden die Probleme und Defizite klassischer Process Mining-Verfahren bei der Anwendung auf agile, personenbezogene Prozesse herausgestellt.

2.4.1 α-Algorithmus

Der α-Algorithmus ist eines der ersten Process Mining-Verfahren, das Nebenläufigkeit von Aktivitäten erkennen konnte und gilt als eine gute Einführung in Process Mining-Techniken Der Algorithmus ist relativ einfach zu verstehen und viele der Ideen werden in zukünftigen Methoden wiederverwendet. Der α-Algorithmus durchsucht das gegebene Ereignisprotokoll nach bestimmten (Kontrollfluss-)Mustern zwischen Aktivitäten und bildet schließlich, auf Basis der abgeleiteten Muster, das Verhalten des Prozesses als **Petri-Netz** ab. Im Rahmen dieser Arbeit wird die Prozessmodellierung mit Petri-Netzen nicht weiter verwendet, weshalb wir für weitere Details auf [132] verweisen. Wenn eine Aktivität a beispielsweise mindestens einmal direkt vor Aktivität b durchgeführt wird, jedoch nie b direkt vor a, dann folgert der Algorithmus daraus, dass eine kausale Abhängigkeit zwischen a und b besteht. Diese Abhängigkeit wird im resultierenden Petri-Netz als Stelle zwischen den Aktivitäten a und b abgebildet.

Petri-Netz als Zielsprache des α-Agorithmus

Tabelle 3: Log des Prozesses aus Sicht des α-Algorithmus

Fall	Zeitstempel	Aktivität	Abkürzung
SS-Riga2013	2013-07-06T15:31	Antrag stellen	a
	2013-07-07T16:28	Antrag prüfen	b
	...	...	...
	2013-07-09T10:28	Antrag genehmigen	c
	2013-07-10T09:45	Unterkunft buchen	d
	2013-07-10T10:29	Antrag archivieren	e
SJ-Muc2014	2014-11-06T15:06	Unterkunft buchen	d
	2014-11-08T15:59	Antrag stellen	a
	2014-11-09T09:22	Antrag prüfen	b
	...	...	...
	2014-11-16T10:24	Antrag genehmigen	c
	2014-11-17T12:59	Antrag archivieren	e
...	...	...	...

Der Algorithmus kennt keine verschiedenen Ereignistypen, d.h. Start oder Beendigung einer Aktivität werden nicht unterschieden. Wir betrachten hier der Einfachheit halber daher nur die *Complete*-Ereignisse des Ereignisprotokolls. Wie bereits erwähnt wird lediglich der Ablauf des Prozesses ohne Einbeziehung der Ressourcen analysiert. Aus der potentiell vorhandenen Information **c**(a,p) eines Ereignisses verwendet der Algorithmus daher lediglich die Information a. Der α-Algorithmus verwendet daher nur die in Tabelle 3 dargestellte Information des ursprünglichen Ereignisprotokolls aus Tabelle 2. Insgesamt kennt der Algorithmus vier verschiedene Relationen zwischen Aktivitäten [7]:

Relationen des α-Algorithmus

- $a > b$, wenn es eine Spur $\sigma = \langle e_1, e_2, e_3, ..., e_n \rangle$ in Φ gibt und $e_i = a$ und $e_{i+1} = b$ gilt.
- $a \rightarrow b$, wenn sowohl $a > b$ als auch $b \not> a$ gilt.

- $a\#b$, wenn $a \not> b$ und $b \not> a$ gilt.
- $a||b$, wenn $a > b$ und $b > a$ gilt.

Sei nun das exemplarische Ereignisprotokoll $\Phi = \{\langle a, b, ..., c, d, e\rangle, \langle d, a, b, ..., c, e\rangle\}$ aus Tabelle 3 gegeben. Für dieses Ereignisprotokoll ergeben sich folgende Relationen:

- $>= \{(a, b), (b, ...), (..., c), (c, d), (d, e), (d, a), (c, e)\}$
- $\rightarrow = \{(a, b), (b, ...), (..., c), (c, d), (d, e), (d, a), (c, e)\}$
- $\# = \{(a, a), (a, c), (b, b), (a, e), (e, a), (c, a), (b, e), (e, b), (b, d), (d, b), (c, c), (d, d), (e, e), (a, ...), (..., a), (d, ...), (..., d), (e, ...), (..., e)\}$
- $|| = \emptyset$

Die Relation $>$ beinhaltet alle Paare von Aktivitäten, die mindestens einmal direkt aufeinanderfolgen. Beispielsweise gilt $c > d$, weil Aktvität d in der ersten Spur direkt auf c folgt. $d > c$ gilt jedoch nicht, da c in keiner Spur direkt auf d folgt. In Relation $\rightarrow$ sind alle Paare von Aktivitäten enthalten, die in einer kausalen Relation stehen. Es gilt beispielsweise $c \rightarrow d$, da d zwar mindestens einmal direkt auf c folgt, jedoch nie c auf d. Der Algorithmus folgert daraus eine kausale Abhängigkeit zwischen c und d. Nebenläufige bzw. parallele Aktivitäten sind in der $||$-Relation enthalten. Im Beispiel sind keine parallelen Aktivitäten enthalten. Folgen zwei Aktivitäten nie direkt aufeinander, befinden sie sich in der #-Relation, wie beispielsweise b und e, da $b \not> e$ und $e \not> b$ gilt. Auf diese Weise wird für jedes mögliche Paar an Aktivitäten eine Relation entdeckt.

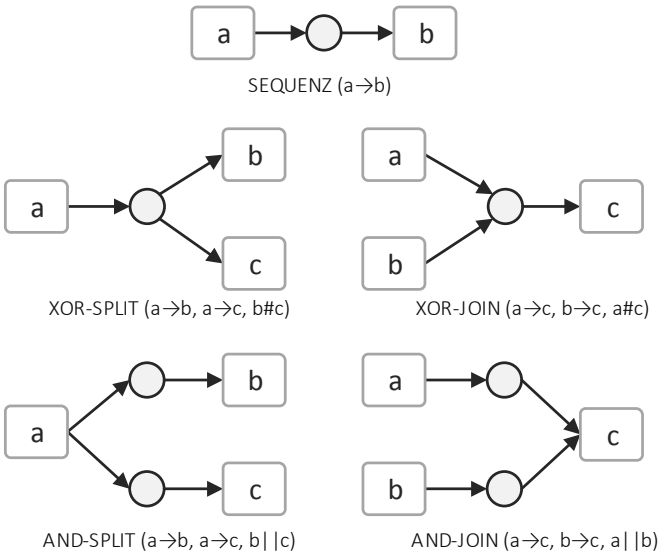


Abb. 13: Kontrollflussmuster des α-Algorithmus [10]

Abgeleitete Kontrollfluss-muster

Auf Basis dieser Menge an Relationen werden anschließend Kontrollfluss-

muster in der Petri-Netz-Notation abgeleitet, welche in Abbildung 13 abgebildet sind. Der Algorithmus ist daher in der Lage die Kontrollflussmuster *Sequenz*, *exklusive Verzweigung* bzw. *exklusive Zusammenführung* und *parallele Verzweigung* bzw. *parallele Zusammenführung* zu erkennen. Aus $d \rightarrow a$, $d \rightarrow e$ und $a\#e$ wird im Beispiel daher eine exklusive Verzweigung abgeleitet. Aus dem gegebenen Beispiel-Ereignisprotokoll wird durch Anwendung des Algorithmus das Petri-Netz aus Abbildung 14 extrahiert.

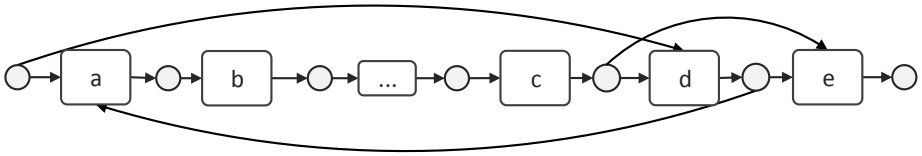


Abb. 14: Abgeleitetes Petri-Netz des Beispiel-Protokoll

2.4.2 Probleme klassischer Process Mining-Verfahren

Der prozedurale Modellierungsansatz ist für agile Prozesse eher ungeeignet [27], [80]. Konsequenterweise steht der erfolgreichen Anwendung von prozeduralen Process Mining-Techniken im Kontext agiler, personenbezogener Prozesse eine Reihe von Problemen entgegen, welche im folgenden Abschnitt erläutert werden.

Komplexität extrahierter prozeduraler Prozessmodelle

Schwer interpretierbare Modelle

Wendet man prozedurale Verfahren auf Ereignisprotokolle von agilen Prozessen an, so sind die Ergebnisse meist **sehr komplex**, verwirrend und schwer zu interpretieren. Die resultierenden Modelle beinhalten häufig eine unüberschaubare Anzahl an Verbindungen zwischen den Aktivitäten, welche die vielen verschiedenen Ausführungsmöglichkeiten abbilden. Modelle, die als Graphen visualisiert sind, werden schon bei relativ wenigen Knoten und Kanten schell unüberschaubar und überfordern das menschliche Wahrnehmungsvermögen [55]. Wird ein Prozess häufig auf unterschiedliche Weise durchgeführt und besitzt demzufolge viele verschiedene Ablaufmöglichkeiten, so ist es typischerweise nicht möglich, ein sinnvolles prozedurales Prozessmodell zu extrahieren [9].

Das Modell aus Abbildung 14, das durch Anwendung des α-Algorithmus auf das einfache Ereignisprotokoll aus Tabelle 3 entstanden ist, unterstreicht das. Das Modell zeigt, dass der analysierte Prozess in einer Instanz mit einer Aktivität begonnen wird, welche in einer anderen Instanz erst gegen Ende des Prozesses auftritt (z.B. “Unterkunft buchen”, Abkürzung “d”). Einerseits existiert ein Sequenzflusspfeil direkt vom Startknoten zur Aktivität “d”, andererseits kann Aktivität “d” auch durch Durchlaufen des gesamten Modells (von “a” über “b” zu “c” etc.) erreicht werden. Dies ist auf die Agilität des Prozesses und die gegebenen Freiräume für die Benutzer zurückzuführen. Bei der Ana-

lyse mit dem α-Algorithmus führt diese Tatsache, trotz der relativ geringen Anzahl an Aktivitäten, schnell zu einem tendenziell unüberschaubaren Modell, da zahlreiche explizite Sequenzflusslinien von der einen Seite zur anderen Seite des Modells (und zurück) erzeugt werden.

Spaghetti-Modelle

Aufgrund der resultierenden Unüberschaubarkeit werden diese Modelle als **„Spaghetti-Modelle“** bezeichnet [58]. Ein Beispiel eines “Spaghetti-Modells” zeigt Abbildung 15. Die Abbildung spiegelt nur rund 20% des gesamten Prozesses wider. Dieses prozedurale Modell wurde aus dem Ereignisprotokoll eines agilen Prozesses mittels des HeuristicsMiner [137] extrahiert, einem klassischen prozeduralen Process Mining-Algorithmus. Die übermäßige Fülle an Kanten erschwert es aus derartigen Modellen nützliche Informationen zu extrahieren. Es ist wichtig zu erwähnen, dass das Modell aus Abbildung 15 nicht falsch ist – es bildet den im Protokoll aufgezeichneten, zu Grunde liegenden Prozess nur unvorteilhaft ab [58].

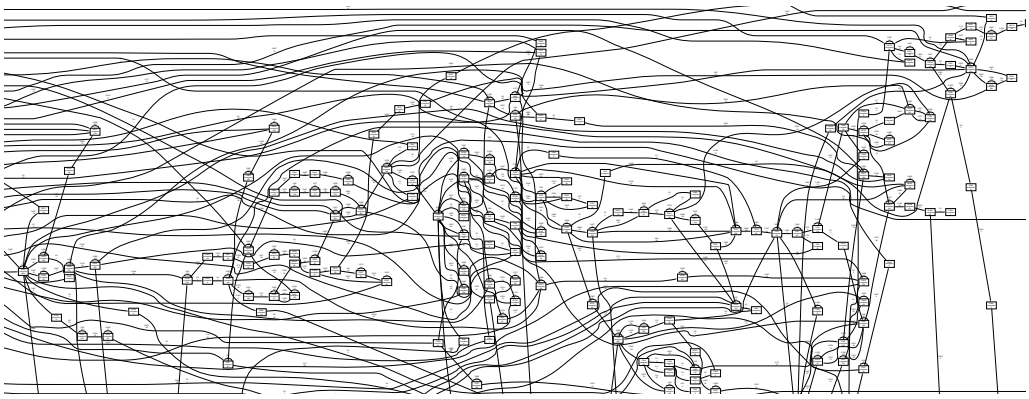


Abb. 15: Auszug eines “Spaghetti-Modells” [58]

FuzzyMining

Der *FuzzyMiner* [57], [58] stellt besondere Art des prozeduralen Process Mining dar. Das Verfahren ermöglicht Benutzern die Visualisierung des extrahierten Prozessmodells auf Basis von abstrahierenden Metriken interaktiv zu verändern. Ziel des Verfahrens ist es, die Lesbarkeit und Verständlichkeit von “Spaghetti-Modellen” zu verbessern. Der FuzzyMiner verwendet zwei grundsätzliche Metriken um Prozessmodelle zu vereinfachen, namentlich **Signifikanz** (*engl. significance*) und **Korrelation** (*engl. correlation*). Durch die interaktive Angabe von Werten für diese Metriken können Prozessanalysten entscheiden, ein detailliertes Modell anzuzeigen, was zwar alle möglichen Pfade enthält, jedoch eher unübersichtlich ist. Andererseits können durch einen niedrigen Signifikanz-Wert eher irrelevante, d.h. weniger häufig begangene Pfade, aus dem Modell gefiltert werden (*engl. edge cut-off*). Durch die Korrelationsfunktion des Verfahrens können Teilbereiche des Prozessmodells zu Subprozessen aggregiert werden. Auch dies erhöht die Übersichtlichkeit des Modells.

Abbildung 16 [58] zeigt ein extrahiertes Modell vor und nach der Anwendung einer auf dem Signifikanz-Wert basierten Filterung von Kanten durch

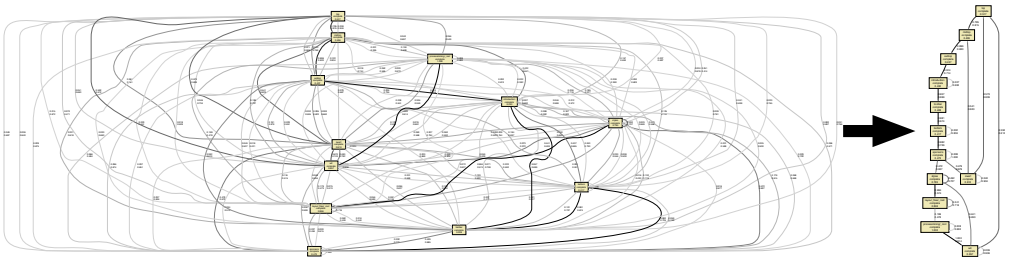


Abb. 16: Fuzzy Modell mit Anwendung der Kantenfilterung

die Fuzzy Mining-Implementierung des ProM Frameworks (Version 6.3)[4]. Zu beachten ist jedoch, dass es sich bei dem extrahierten Modell um kein ausführbares Prozessmodell handelt, sondern lediglich um eine graphische Visualisierung. Wird das Modell mit allen Details und ohne Filterung von Pfaden angezeigt, so entsteht wieder ein unüberschaubares "Spaghetti-Modell". Die verbesserte Lesbarkeit des Modells wird somit letztendlich durch "Löschen" von (potentiell wichtigen) Informationen erreicht. Das dargestellte Prozessmodell repräsentiert ist nicht mehr vollständig, sondern repräsentiert lediglich noch häufiges Verhalten.

Unzureichende Analyse der organisatorischen Perspektive

Ein Modell bildet das aus der Datenbasis extrahierte Wissen in einer bestimmten Sprache bzw. Notation ab, einer sog. **Modellierungssprache** [140]. Sämtliche abgeleitete Muster und Modelle werden in der gewählten Modellierungssprache dargestellt. Sowohl Data Mining- als auch Process Mining-Verfahren können nur innerhalb der formalen Einschränkungen der gewählten Modellierungssprache agieren. Die Wahl einer bestimmten Modellierungssprache beeinflusst in großem Maße den eigentlichen Process Mining-Prozess [6] und bringt häufig eine Reihe impliziter Annahmen mit sich, welche den Suchraum einschränken [10]. Zusammenhänge und Muster, die in der gewählten Modellierungssprache nicht darstellbar sind, können auch nicht abgeleitet, d.h. entdeckt werden. Jegliche Art von Process Mining-Verfahren unterliegt in unterschiedlichem Maß dieser sog. **Darstellungsausrichtung** (*engl. representational bias*) [10]. Die Darstellungsausrichtung legt den Suchraum des Process Mining-Prozesses fest und schränkt dadurch die Ausdrucksfähigkeit des abgeleiteten Modells ein.

Formale Einschränkungen der gewählten Modellierungssprache

Darstellungsausrichtung von Process Mining

Prozedurale Process Mining-Techniken, wie der im vorherigen Abschnitt beschriebene α-Algorithmus, verwenden als Zielsprache Petri-Netze oder eine Teilmenge der BPMN-Notation [47], deren Sprachkonstrukte sich exklusiv auf die verhaltensorientierte Perspektive, d.h. den Kontrollfluss beziehen. Kürzlich veröffentlichte Forschungsarbeiten verwenden **gefärbte Petri-Netze** (*engl. coloured petri nets)*, einen erweiterten Petri-Netz Formalismus, zur zusätzlichen Modellierung von Datenflüssen [69], [75]. Abbildung 13 zeigt die Kontrollflussmuster, die vom klassischen α-Algorithmus berücksichtigt werden. Da die ver-

4 Verfügbar unter www.promtools.org

Fehlende Betrachtung der organisatorischen Perspektive

wendeten Zielsprachen die beteiligten Akteure und Ressourcen (Organisatorische Perspektive) nicht einbeziehen, werden diese Perspektiven auch in den extrahierten Modellen nicht berücksichtigt [80]. In agilen Prozessen haben prozessbeteiligte Akteure jedoch einen signifikanten Einfluss auf den tatsächlichen Prozessablauf [80]. Im Fall von agilen, personenbezogenen Prozessen muss das gegebene Ereignisprotokoll daher auch aus einer organisatorischen Perspektive analysiert werden.

Klassische organisatorische Mining-Verfahren

Neben Verfahren, welche die verhaltensorientierte Perspektive betrachten, existieren eine Reihe von klassischen Process Mining-Techniken, die Ereignisprotokolle aus der **organisatorischen Perspektive** analysieren [125]. Ausgangspunkt für eine Analyse der organisatorischen Perspektive ist das Ressourcen-Attribut (*org:resource* in XES-Dokumenten), das in den meisten Ereignisprotokollen aufgezeichnet ist [10]. Klassische, organisatorische Mining-Verfahren leiten hauptsächlich Beziehungen zwischen Aktivitäten und Ressourcen ab, d.h. beispielsweise die Menge an Personen, die eine bestimmte Aktivität durchgeführt hat. Mit Hilfe von agglomerativen Clustering-Verfahren können auf diese Weise Gruppen oder Rollen automatisiert abgeleitet werden [125]. Durch Anwendung des *Organizational Mining*-Algorithmus [125] von Song aus dem Jahr 2008 kann das Modell aus Abbildung 14 durch Ressourcen-Information erweitert werden. Die Ergebnisse der Analyse der organisatorischen Perspektive können dem bereits abgeleiteten prozeduralen Modell *nachträglich* hinzugefügt werden [10]. Angewendet auf das Beispiel-Ereignisprotokoll aus Tabelle 2 entsteht das in Abbildung 17 dargestellte, durch Ressourcen-Information erweiterte Prozessmodell.

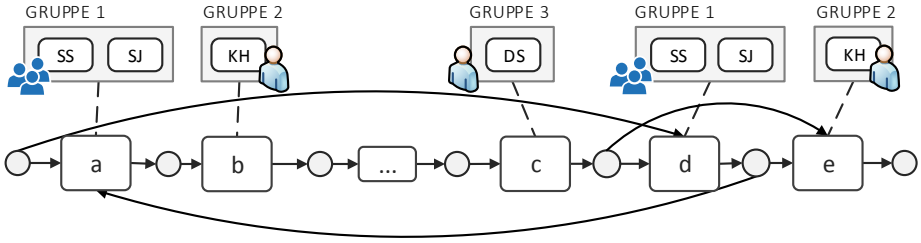


Abb. 17: Modell erweitert durch Zuweisungsmuster

Unentdeckte perspektivenübergreifende Muster

Das Modell zeigt nun beispielsweise, dass die Aktivitäten "Antrag stellen" (a) und "Unterkunft buchen" (d) von den Identitäten "SS" und "SJ"' durchgeführt werden. Dies ist eine nützliche Information, erfasst jedoch nicht das gesamte Spektrum an häufig auftretenden organisatorischen Mustern in Prozessen. Häufig treten komplexe, unter Umständen auch **perspektivenübergreifende Zusammenhänge** auf, die durch klassische Process Mining-Verfahren nicht entdeckt werden können. Bereits bei diesem einfachen Beispiel lassen sich zahlreiche Fragestellung formulieren, welche durch das abgeleitete Modell nicht beantwortet werden können:

- **Verknüpfung von Zuständigkeiten:** Werden die Aktivitäten "Antrag stellen" (*a*) und "Unterkunft buchen" (*d*) stets von der gleichen Person durch-

geführt? Das Modell agglomeriert sämtliche durchführende Personen einer Aktivität. Auf diese Weise ist nicht mehr ersichtlich, ob in einer Instanz, in der a von "SS" durchgeführt wird, auch d von "SS" bearbeitet wird.

- **Trennung von Zuständigkeiten:** Auch die invertierte Fragestellung bleibt durch das Modell entsprechend unbeantwortet. Es ist nicht nachvollziehbar, ob "a" und "d" immer von unterschiedlichen Personen oder der gleichen Person zu bearbeiten sind.

- **Organisatorische Beziehungen:** Stehen die Personen, die gewisse Aktivitäten durchgeführt haben, in einer bestimmten organisatorischen Beziehung zueinander? Können Personen in jeder Rolle einen Antrag genehmigen?

- **Einfluss der organisatorischen Perspektive auf den Kontrollfluss:** In vielen Fällen ist der Kontrollfluss, d.h. die verhaltensorientierte Perspektive, abhängig von der durchführenden Person und deren Rolle bzw. deren organisatorischer Gruppierung. Diese Information geht durch die strikte Trennung der Analyse der verhaltensorientierten und der organisatorischen Perspektive verloren. Im Modell ist beispielsweise nicht klar, ob sowohl Person "SJ" als auch Person "SS" eine Unterkunft buchen (d) können, ohne einen Antrag gestellt (a) zu haben.

Unpräzise Modelle bei klassischen Verfahren

Diese Beispiele zeigen, dass viele komplexe, jedoch häufig auftretende organisationsbezogene Prozessmuster existieren, welche durch klassische Process Mining-Techniken nicht abgeleitet werden können. Als Folge davon werden unpräzise Modelle ermittelt, welche nur eine Annäherung bzw. Approximation an die komplexen, realen Sachverhalte darstellen [66].

2.5 ZUSAMMENFASSUNG

In diesem Kapitel werden die Grundlagen der automatisierten Modellierung von Prozessen erläutert. In Organisationen und Unternehmen werden heutzutage Daten in enormen Mengen digital aufgezeichnet. **Data Mining**-Verfahren sind Computer-gestützte Methoden zur (semi-)automatisierten Analyse dieser Daten. Im Gegensatz zu traditionellen Analysemethoden, wie manueller Sichtung und Interviews, ist die IT-gestützte Datenanalyse wesentlich schneller, nahezu frei von subjektiver Interpretation und daher wesentlich **objektiver**.

Der Begriff **Process Mining** bezeichnet die Anwendung von Data Mining Methoden im Kontext von Prozessen. Process Mining verwendet die Daten, die während der IT-gestützten Durchführung von Prozessen protokolliert werden, zur Ableitung von Prozessmodellen. Ausgangspunkt und Datengrundlage für Process Mining-Verfahren sind digitale, **ereignisorientierte Prozessausführungsprotokolle** (engl. Process Execution Logs), kurz **Logs**. Ein Ereig-

nisprotokoll enthält typischerweise Ereignisaufzeichnungen, die genau zu einem Prozess gehören und genau einer Prozessinstanz, d.h. einem bestimmten Fall des Prozesses, zuzuordnen sind. Das Standardformat zur Speicherung und zum Austausch von Ereignisprotokollen ist das XML-basierte ***eXtensible Event Stream***, kurz *XES*.

Während des letzten Jahrzehnts wurden zahlreiche Process Mining-Verfahren zur automatisierten Modellierung von Prozessen entwickelt. Der Großteil der Verfahren strebt dabei die Extraktion eines Prozessmodells an (*engl. Process Discovery*). Klassische Process Mining Techniken erzeugen Prozessmodelle in prozeduralen Modellierungssprachen wie Petri-Netzen. Exemplarisch für prozedurale Verfahren wurde der **α-Algorithmus** erläutert, welcher lediglich die **verhaltensorientierte Perspektive** betrachtet. Wendet man prozedurale Verfahren auf die Ereignisprotokolle von agilen Prozessen an, so sind die Ergebnisse meist **sehr komplex** und schwer zu interpretieren. Die resultierenden Modelle beinhalten eine unüberschaubare Anzahl an Verbindungen zwischen den Aktivitäten, welche die vielen verschiedenen Ausführungsmöglichkeiten abbilden.

Des Weiteren berücksichtigen klassische Verfahren die organisatorische Perspektive von Prozessen nur unzureichend. Dies liegt hauptsächlich in der Wahl der Zielsprache des Mining-Verfahrens begründet. Process Mining-Verfahren können nur innerhalb der formalen Einschränkungen der gewählten Modellierungssprache agieren. Zusammenhänge, die in der gewählten Modellierungssprache nicht darstellbar sind, können auch nicht abgeleitet werden. Diese **Darstellungsausrichtung** von klassischen Process Mining-Verfahren verhindert, dass komplexere organisatorische Zuweisungsregeln, wie eine Verknüpfung oder Trennung von Zuständigkeiten, nicht abgeleitet werden können. Auch **perspektivenübergreifende Zusammenhänge**, wie der Einfluss der organisatorischen Perspektive auf den Kontrollfluss, kann daher nicht entdeckt werden.

3 MODELLIERUNGSSPRACHEN FÜR AGILE PROZESSE

INHALT

Aussagekraft von Resultaten abhängig von Zielsprache

Im vorherigen Abschnitt wird gezeigt, dass die Qualität bzw. Verständlichkeit und Aussagekraft der Resultate von Process Mining in großem Maße von der verwendeten Zielsprache bzw. Notation abhängig sind. Nur Muster, die in der gewählten Sprache abbildbar, d.h. modellierbar sind, können auch entdeckt bzw. extrahiert werden. Daher kommt der Auswahl der Zielsprache eine besondere Bedeutung zu. Um eine passende Zielsprache für die automatisierte Prozessmodellerzeugung agiler, personenbezogener Prozesse auszuwählen, rekapitulieren wir zuvor noch einmal deren Eigenschaften und deren Anforderungen an die Modellierung, wie bereits in Kapitel 1 herausgestellt wurde.

Eigenschaften agiler Prozesse

Besonders variantenreich

- Akteure besitzen in agilen Prozessen deutlich mehr **Entscheidungsfreiheit** bei der Wahl der durchzuführenden Aktivitäten, als auch deren Ausführungsreihenfolge. Modelle müssen daher **besonders variantenreiche Prozesse verständlich** abbilden können (A1).

Verpflichtende und empfohlene Aktionen

- Um beteiligten Akteuren trotz der gegebenen Entscheidungsfreiheit ausreichende Unterstützung zu bieten, ist die **Unterscheidung von verpflichtenden** und lediglich **empfohlenen Aktionen** im Modell sinnvoll (A2).

Komplexe organisatorische Muster

- Agile, personenbezogene Prozesse sind in großem Maße abhängig von menschlichen Akteuren und deren Expertenwissen. Aufgaben werden

Personen, Rollen oder Benutzergruppen auf Basis komplexer Zuweisungsregeln (z.B. Verknüpfung/Trennung von Zuständigkeiten) zugewiesen. Diese **organisatorische Perspektive** des Prozesses muss sich adäquat in Modellen abbilden lassen (A3).

Perspektivenübergreifende Muster

- Neben der Aufgabenzuweisung hat die organisatorische Perspektive in derartigen Prozessen auch großen Einfluss auf den Prozessablauf, d.h. den Kontrollfluss bzw. die verhaltensorientierte Perspektive. Es treten **perspektivenübergreifende** Zusammenhänge auf, die in Prozessmodellen abgebildet werden müssen (A4).

Erläuterung anhand Beispielprozess

Anhand dieser Anforderungen werden im folgenden Kapitel existierende, ausführbare Prozessmodellierungssprachen bewertet und eine passende Zielsprache ausgewählt. Zur Erläuterung der Auswahl einer adäquaten Prozessmodellierungssprache betrachten wir den Ausschnitt des exemplarischen universitären Dienstreiseabwicklungsprozesses aus Kap. 1.2.3. Anhand dieses Prozesses können die Anforderungen an die Modellierung gut erläutert werden. Die erläuterten Zusammenhänge finden sich jedoch in vielen anderen realen Prozesse wieder, wie in Forschung- und Entwicklungsprozessen oder in klinischen Prozessen. Die zuvor genannten Eigenschaften agiler, personenbezogener Prozesse sind bereits in diesem einfachen Beispiel enthalten. Eine adäquate Zielsprache muss es ermöglichen, all die genannten Zusammenhänge in einem Prozessmodell abbilden zu können. Hierzu werden zuerst die ausgewählten Sprachen beschrieben, deren Funktionalität und theoretischer Hintergrund beleuchtet und anschließend versucht den agilen Beispielprozess in der entsprechenden Sprache abzubilden. Auf Basis dessen wird folglich deren Eignung evaluiert.

3.1 PROZEDURALE SPRACHEN MIT ORGANISATORISCHEN ERWEITERUNGEN

Prozedurale Sprachen

Wie bereits erwähnt, lässt sich das Spektrum an Prozessmodellierungssprachen in prozedurale und regelbasierte Modellierungssprachen aufteilen. Prozedurale Modellierungssprachen wie BPMN, YAWL, EPKs oder Petri-Netze machen die möglichen Zustände, in denen sich der modellierte Prozess befinden kann, und vor allem die Übergänge zwischen ihnen explizit. Zustände und Übergänge bilden also explizite Elemente des Modells und stehen damit im Vordergrund. Wie in der Einführung dieser Arbeit bereits gezeigt wird, eignen sich aufgrund dessen prozedurale Sprachen eher weniger zur Modellierung agiler Prozesse [48]. Auf Basis prozeduraler Sprachen lässt sich A1 nur schwer erfüllen, da die Entscheidungsfreiheit, welche in agilen Prozessen benötigt wird, nur durch eine Vielzahl an expliziten Sequenzflüssen realisiert werden kann. Auf diese Weise werden Modelle schnell schwer leserlich und unverständlich. Da viele klassische Process Mining-Methoden auf prozeduralen Sprachen basieren, sind

Viele explizite Sequenzflusslinien

diese folglich für agile Prozesse eher ungeeignet.

⊘ **A1:** Schwer verständliche, flussorientierte Modelle im Kontext agiler Prozesse

In prozeduralen Modellen ist bislang keine Unterscheidung in verpflichtende und empfohlene Aktionen möglich. Dies liegt im Prinzip der Modellierung begründet.

⊘ **A2:** Keine Unterscheidung verpflichtender und lediglich empfohlener Aktionen möglich

Organisatorische Ausdrucksspra-chen

Es existieren Ausdruckssprachen, wie die *Resource Assignment Language (RAL)* [32], welche in prozeduralen Sprachen eingebunden werden können. RAL Ausdrücke können als formaler Ausdruck in das Attribut-Feld von Aktivitäten in BPMN-Modellen eingebettet werden. Die Ausdrucksstärke von RAL reicht von der Definition einfacher Zuweisungsregeln, wie der Zuweisung von Aufgaben zu konkreten Personen oder Rollen, bis hin zu komplexen Zuweisungsregeln, wie der Verknüpfung und Trennung von Zuständigkeiten. Auf diese Weise lassen sich komplexe organisatorische Zuweisungsregeln in prozeduralen Sprachen wie BPMN abbilden.

✓ **A3:** Modellierung komplexer organisatorischer Zusammenhänge möglich

Keine perspektiven-übergreifenden Muster

Aktuelle Vertreter der prozeduralen Modellierung wie BPMN setzen einen konsistenten Kontrollfluss voraus, ordnen alle weiteren Perspektiven diesem unter und behandeln diese getrennt voneinander. Eine perspektivenübergreifende Modellierung im Sinne der Anforderung ist somit nicht möglich. Jede Anforderung muss zunächst in ihre jeweiligen Perspektiven zerlegt werden, bevor sie abgebildet werden kann.

⊘ **A4:** Keine perspektivenübergreifende Modellierung möglich

3.2 REGELBASIERTE PROZESSMODELLIERUNGSSPRACHEN

Regelbasierte Modellierung

Sobald die möglichen Zustände eines Prozesses und die Übergänge zwischen ihnen nicht mehr expliziter Teil des Prozessmodells sind, handelt es sich nicht mehr um ein prozedurales Prozessmodell. Die direkten Vorgänger und Nachfolger eines bestimmten Zustands lassen sich dann nicht mehr wie zum Beispiel in einem Petri-Netz über Transitionen erreichen sondern ergeben sich durch Auswertung einer Menge von definierten Regeln. Es besteht kein expliziter Zusammenhang mehr zwischen dem Modell und seinem Verhalten. Im Folgenden werden die bestehenden Ansätze zur regelbasierten Modellierung *ausführbarer Prozessmodelle* beleuchtet, die sich besser zur Modellierung agiler Prozesse eig-

nen. Diese Evaluation existierender Modellierungssprachen ist bereits in einer Publikation des Autors veröffentlicht [114].

3.2.1 Declare Rahmenwerk

Declare ist ein Rahmenwerk zur Modellierung und Ausführung von regelbasierten Prozessmodellen [101].

Beschreibung und Evaluation

Regelschablonen

ConDec und DecSerFlow

Lineare Temporale Logik

Innerhalb des Rahmenwerks lassen sich Regelschablonen definieren und zu einer Sprache zusammenfassen. Die beiden häufig genannten Sprachen sind *ConDec* [103] und *DecSerFlow* [133]. Um die Modelle ausführen zu können, müssen die Regeln auf einen interpretierbaren Formalismus abgebildet werden. Bisher werden im Rahmen von Declare zwei Formalismen vorgeschlagen: die *Lineare Temporale Logik (LTL)* und das *Ereigniskalkül* (*engl. event calculus, kurz EC*) nach Kowalski und Sergot [88]. Für jede Regelschablone muss eine Entsprechung im jeweiligen Formalismus hinterlegt werden. Die Prozessmodelle können so in eine validierbare und ausführbare Form übersetzt werden. Ein Beispiel ist die Regel "um Aktivität b durchführen zu können, muss irgendwann zuvor Aktivität a durchgeführt worden sein". Diese Regel wird häufig als Makro *precedence(a, b)* ausgedrückt. Als grafische Repräsentation dieses Makros ist ein Pfeil mit einem Punkt am Ende vorgesehen.

Die Abbildung dieser Regel auf einen LTL-Ausdruck ist durch $(\neg B)WA$ gegeben. Mit Hilfe der LTL lassen sich Aussagen über Folgen von Zuständen formulieren. Das Symbol *W* sagt aus, dass der Ausdruck auf der einen Seite des Symbols so lange gelten muss, bis die andere Seite des Symbols erfüllt ist. Die Regel besagt also, dass B so lange nicht existieren darf, bis A existiert. Die LTL-Formel muss zur Interpretation wiederum in einen Automaten überführt werden, der dann durchlaufen werden kann und die erlaubten Zustandsübergänge enthält [101]. Declare ist ein regelbasiertes Modellierungsframework, das entworfen wurde, um agile, variantenreiche Prozesse übersichtlich abbilden zu können und um Benutzern während der Durchführung mehr Entscheidungsfreiheit zu gewähren.

Übersichtliche Abbildung agiler Prozesse

✓**A1:** Übersichtliche Abbildung agiler Prozesse in regelbasierten Modellen

Optionale Regeln

Das Declare Framework unterstützt sowohl verpflichtende als auch sog. *optionale* Regeln [102]. Während der Ausführung müssen verpflichtende Regeln erfüllt bleiben. Im Fall von optionalen Regeln können Benutzer entscheiden, ob sie die Regel einhalten oder verletzen wollen. Wird eine optionale Regel durch Aktionen des Benutzers verletzt, so wird ihm eine Warnung angezeigt. Anschließend kann entschieden werden mit der Aktion fortzufahren oder der Empfehlung des Systems zu folgen. Declare ermöglicht daher die Unterscheidung von verpflichtenden und lediglich empfohlenen Aktionen.

✓ **A2:** Unterscheidung von verpflichtenden und empfohlenen Aktionen

Die organisatorische Perspektive wird in Declare nur rudimentär betrachtet. Unabhängig von spezifischen Modellen können im System einfache Organisationsstrukturen definiert werden, d.h. Benutzer und deren Rollen [101]. Eine Definition von beliebigen Beziehungen zwischen organisatorischen Elementen ist nicht möglich. Während der Spezifikation von konkreten Modellen können Aktivitäten eine oder mehrere Rollen zugewiesen werden. Während der Ausführung eines Modells wird eine Aktivität schließlich den Benutzern der zugewiesenen Rollen angeboten. Auf diese Weise ermöglicht Declare lediglich die Modellierung einer rollenbasierten Aufgabenzuweisung. Komplexere organisatorische Zusammenhänge, wie eine Verknüpfung bzw. Trennung von Zuständigkeiten oder eine Aufgabenzuweisung auf Basis organisatorischer Beziehungen, können daher nicht modelliert werden.

Einfache organisatorische Muster

⊘ **A3:** Unzureichende Unterstützung der organisatorischen Perspektive

Regeln in Declare setzen lediglich den Beginn von Aktivitäten zeitlich zueinander in Beziehung. Die Regeln können im Ganzen wiederum von Bedingungen abhängen. Die Regel *precedence(a, b)* mit der Bedingung „x < 3“ fordert, dass b durchgeführt werden kann, wenn irgendwann zuvor a durchgeführt wurde, falls der Wert der Variable x kleiner drei ist. Deshalb können Regeln andere Perspektiven außer der verhaltensorientieren Perspektive nicht beeinflussen. Bedingungen einer Regeln können sich außerdem nicht auf die organisatorische Perspektive beziehen. Eine perspektivenübergreifende Modellierung im Sinne der Anforderungen ist daher nicht möglich.

Keine perspektivenübergreifenden Muster

⊘ **A4:** Keine perspektivenübergreifende Modellierung möglich

Agiler, personenbezogener Beispielprozess in Declare

Abbildung 18 zeigt das Modell des Beispielprozesses aus Abschnitt 1.2.3 in Declare (ConDec-Sprache). Die verhaltensorientierte Perspektive des Prozesses kann mit *precedence*-Regeln abgebildet werden. Zwischen den Aktivitäten “Antrag stellen” und “Antrag genehmigen” ist die *precedence*-Regel verpflichtend (durchgezogener Pfeil mit Punkt), wohingegen sie zwischen “Antrag genehmigen” und “Unterkunft buchen” optional ist (gestrichelter Pfeil mit Punkt). Die Empfehlung einen Antrag vor der Buchung genehmigen zu lassen kann demnach in Declare abgebildet werden. Die beiden Nummern auf den Aktivitäten zeigen an, dass diese exakt einmal durchgeführt werden müssen, damit eine Prozessinstanz beendet wird. Rollen und Ressourcenzuweisungen sind nicht Teil der graphischen Notation können jedoch definiert werden. So können die Aktivitäten “Antrag stellen” und “Unterkunft buchen” von allen vorhandenen Personen bearbeitet werden, wohingegen “Antrag genehmigen” nur

Beispielprozess in Declare

durch Personen in der Rolle "Administration" durchgeführt werden darf. Es kann nicht abgebildet werden, dass die Unterkunft vom Antragsteller gebucht werden muss. Darüber hinaus ist es nicht möglich, die verhaltensorientierte Perspektive mit der organisatorischen Perspektive zu kombinieren. Es kann daher nicht modelliert werden, dass Doktoranden verpflichtend einen Antrag stellen müssen, bevor eine Buchung erfolgen kann.

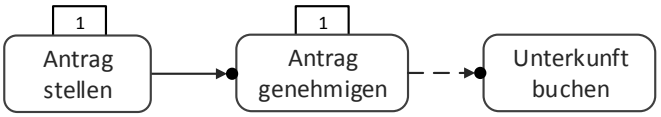


Abb. 18: Agiler Beispielprozess in Declare-Notation (ConDec)

Declare nicht als Zielsprache geeignet

Aufgrund der unzureichenden Unterstützung der organisatorischen Perspektive ist Declare nicht als Zielsprache geeignet. Trotz der aufgezeigten Schwächen ist Declare eine der ersten vollständig ausgearbeiteten regelbasierten Prozessmodellierungssprachen. aufgrund dessen basieren zahlreiche Forschungsarbeiten im Bereich der regelbasierten Prozessmodellierung und Process Mining des letzten Jahrzehnts auf Declare.

3.2.2 Case Management Modelling and Notation (CMMN)

Beschreibung und Evaluation

Fall (Case) als agiler Prozess

Die *Case Management Model and Notation (CMMN)* wurde von der Object Management Group (OMG) Anfang 2013 das erste Mal vorgeschlagen. Die Sprache basiert auf dem *Guard-Stage-Milestone (GSM)*-Modell nach Hull [64], [65]. Die Motivation ist, dass ein *Fall (engl. case)*, anders als ein Prozess, nicht als vorgegebene Sequenz von Aktivitäten abgearbeitet werden kann, weil diese Sequenz nicht ad hoc bekannt ist, sondern erst durch Erfahrung etabliert und außerdem von Fall zu Fall sehr verschieden sein kann [97]. Ein Fall im Sinne der CMMN entspricht also dem, was in dieser Arbeit als agiler Prozess bezeichnet wird. Um Fälle durch IT unterstützen zu können, definiert die CMMN eine regelbasierte Sprache. Die Struktur eines Falls (*Case*) wird im Wesentlichen mittels Aufgaben (*Task*), den Elementen der Fallakte (*CaseFileItem*) und Meilensteinen (*Milestone*) modelliert. Die Teilnehmer werden über Rollen (*Role*) mit den personellen Aufgaben (*HumanTask*) des Falls verknüpft.

Realisierung von Regeln durch Sentries

Regeln werden durch sog. *Sentries* realisiert, bei denen es sich um Tripel aus Ereignis, Bedingung und Aktion (*engl. event condition action, kurz ECA*) handelt. Tritt das definierte Ereignis ein und ist die definierte Bedingung erfüllt, dann wird die entsprechende Aktion ausgelöst. Ereignisse können sich dabei auf Zustandsübergänge von Elementen der Fallakte (*CaseFileItemTransition*) oder von Aufgaben, Meilensteinen oder Ereignissen (*PlanItemTransition*) beziehen. Die Bedingung einer Regel kann sich lediglich auf die Fallakte beziehen. Die resultierende Aktion ist das Betreten oder Verlassen einer Aufgabe, eines Abschnitts oder eines Meilensteins. Agile, besonders variantenreiche Prozesse lassen sich

Regelbasierte Sprache

mit der regelbasierten CMMN daher übersichtlich abbilden.

✓**A1:** Übersichtliche Abbildung agiler Prozesse in CMMN-Modellen

Keine verschiedenen Modalitäten

Prozesse lassen sich in CMMN also durch Regeln einschränken. CMMN unterstützt jedoch keine verschiedenen Modalitäten. Regeln lassen sich nicht hinsichtlich ihrer Qualität klassifizieren und werden somit stets als verpflichtend interpretiert. Nicht verpflichtende, jedoch empfohlene Aktionen lassen sich in CMMN daher nicht abbilden.

⊘ **A2:** Keine Unterscheidung verpflichtender und lediglich empfohlener Aktionen möglich

Feste Rollenzuweisungen

Wie bereits beschrieben, lassen sich Teilnehmer nur über fest vorgegebene Rollen zur Entwurfszeit den Aufgaben zuweisen. In graphischen Visualisierungen wird dies nicht dargestellt. Die organisatorische Perspektive ist in CMMN somit nur ansatzweise abgedeckt.

⊘ **A3:** Unzureichende Unterstützung der organisatorischen Perspektive

Eingeschränkte Perspektivenübergreifende Modellierung

CMMN beinhaltet also die üblichen Perspektiven der Prozessmodellierung. Dabei können aber als Auslöser nur Ereignisse der Verhaltens- und Datenperspektive, als Bedingung nur Zusammenhänge der Datenperspektive und als Konsequenz nur die verhaltensorientierte Perspektive eingesetzt werden. CMMN erlaubt daher nur bedingt eine perspektivenübergreifende Modellierung. Prozesse lassen sich also nicht basierend auf der Organisationsperspektive und nicht hinsichtlich der Daten- und Organisationsperspektive einschränken.

⊘ **A4:** Nur eingeschränkte perspektivenübergreifende Modellierung möglich

Agiler, personenbezogener Beispielprozess in CMMN

Beispielprozess in CMMN

Abbildung 19 zeigt das Modell des agilen Beispielprozess aus Abschnitt 1.2.3 in CMMN. Die zeitliche Abhängigkeit zwischen "Antrag stellen" und "Antrag genehmigen" wird durch den gestrichelten Pfeil mit Diamant modelliert. CMMN kennt nur verpflichtende Regeln, weshalb die Empfehlung den Antrag vor einer Buchung zu stellen nicht ausgedrückt werden kann. Verplichtend auszuführende Aktivitäten werden mit dem äußeren Rahmen durch Linien und einen schwarzen Diamanten verbunden. In CMMN können lediglich die sog. *CaseRoles* definiert werden, welche jedoch nicht Teil des graphischen Modells sind. Es kann demnach abgebildet werden, dass der Schritt "Antrag genehmigen" von einer Person in der Rolle "Administration" durchgeführt werden muss. Auch mit CMMN ist es wiederum nicht möglich, die Verknüpfung der Zuständigkeiten abzubilden. Auch den perspektivenübergreifenden Zusammenhang zwi-

schen “Antrag stellen” und “Unterkunft buchen” kann man mit CMMN nicht modellieren.

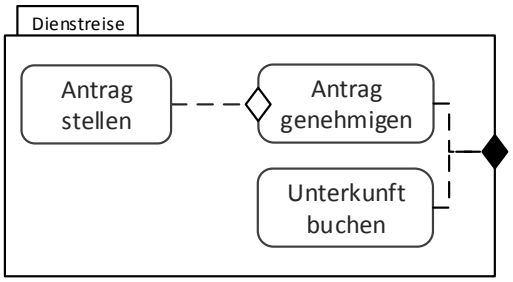


Abb. 19: Agiler Beispielprozess in CMMN

CMMN nicht als Zielsprache geeignet

Die Evaluation hinsichtlich der Anforderungen und das Beispiel zeigen, dass CMMN, aufgrund der fehlenden Modalitäten und der fehlenden Unterstützung der organisatorischen Perspektive, als Zielsprache eher ungeeignet ist. Trotz dieser Einschränkungen kann CMMN als Stand der Technik gesehen werden, was die regelbasierte Prozessmodellierung angeht. Wie auch BPMN kombiniert CMMN eine reduzierte grafische Darstellung in Diagrammen mit einem technisch detaillierten Modell und ist daher gut für die Vermittlung zwischen Fachabteilung und IT geeignet. Ziel zukünftiger Forschungsarbeiten sollte daher sein, die aufgezeigten Schwächen von CMMN aufzuarbeiten.

3.2.3 DCR-Graphen

Beschreibung und Evaluation

DCR-Graphen ähnlich zu Declare

Dynamic Condition Response-Graphen (DCR-Graphen) [63], [90] ist eine deklarative Prozessmodellierungssprache die sehr ähnlich zu Declare ist. Tatsächlich entspricht die graphische Abbildung von Ereignissen und Regeln der von Declare. Die Menge an verwendbaren Regelschablonen von DCR-Graphen ist jedoch deutlich geringer als in Declare. Aktivitäten können durch die Regeltypen *Condition*, *Response*, *Milestone*, *Inclusion* und *Exclusion* miteinander verbunden werden. Nichtsdestotrotz können durch DCR-Graphen Modelle mit der gleichen Aussagekraft wie mit Declare abgebildet werden, da der zu Grunde Formalismus nicht auf LTL basiert. Während Declare Modelle zur Verifikation und Ausführung zuerst in LTL bzw. EC-Regeln transformiert werden müssen, erlauben DCR-Graphen die direkte Ausführung der Modelle auf Basis von Marken innerhalb des Graphen.

Regelbasierte Modelle

✓**A1:** Übersichtliche Abbildung agiler Prozesse in regelbasierten Modellen

Keine Modalitäten

Im Gegensatz zu Declare werden in DCR-Graphen keine verschiedenen Modalitäten unterstützt. Eine Unterscheidung zwischen verpflichtenden und lediglich empfohlenen Aktionen ist deshalb nicht möglich.

⊘ **A2:** Keine Unterscheidung von verpflichtenden und empfohlenen Aktionen

Unzureichende Modellierung des org. Aspekts

Auch die Modellierung der organisatorische Perspektive beschränkt sich, wie in Declare, auf die Definition und Zuweisung von Ressourcen und Rollen zu Aktivitäten [63]. Die Abbildung komplexerer organisatorischer Zuweisungsregeln ist nicht möglich.

⊘ **A3:** Unzureichende Unterstützung der organisatorischen Perspektive

Keine perspektivenübergreifende Modellierung

Da nur die oben genannten Regeltypen zwischen Aktivitäten modelliert werden können, unterstützen DCR-Graphen auch keine perspektivenübergreifende Modellierung. Die verhaltensorientierte Perspektive kann demnach also nicht durch organisatorische Umstände beeinflusst werden.

⊘ **A4:** Keine perspektivenübergreifende Modellierung möglich

Agiler, personenbezogener Beispielprozess als DCR-Graph

Beispielprozess als DCR-Graph

In Abbildung 20 ist der Beispielprozess aus Abschnitt 1.2.3 in der DCR-Graph-Notation dargestellt. Die zeitliche Abhängigkeit zwischen "Antrag stellen" und "Antrag genehmigen" wird durch eine *Condition*-Regel (entspricht der *precedence*-Regel in Declare) abgebildet. Da DCR-Graphen keine unterschiedlichen Regel-Modalitäten unterstützen, kann die Empfehlung einen Antrag vor der Buchung zu stellen nicht modelliert werden. Rollenzuweisungen sind in DCR-Graphen auch graphisch visualisierbar. Das Modell zeigt hier, dass Anträge von der Administration genehmigt werden müssen. Auch mit DCR-Graphen kann keine Verknüpfung von Zuständigkeiten abgebildet werden. Die zeitliche Abhängigkeit zwischen "Antrag stellen" und "Unterkunft buchen" im Fall von Doktoranden kann ebenso mit DCR-Graphen nicht modelliert werden.

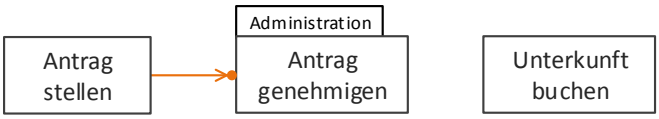


Abb. 20: Agiler Beispielprozess in der DCR-Graph-Notation

DCR-Graphen stellen eine auf Declare basierende weiterentwickelte regelbasierte Prozessmodellierungssprache dar. Sie legt den Fokus vor allem darauf, die Menge an verfügbaren Modellierungskonstrukten übersichtlich zu halten. Dennoch ist es möglich, Modelle mit der Aussagekraft wie in Declare zu konstruieren. Die Evaluation und das Beispiel zeigen, dass die Sprache, vor allem aufgrund der fehlenden Elemente der organisatorischen Perspektive als Zielsprache eher ungeeignet ist.

3.2.4 Weitere regelbasierte Prozessmodellierungssprachen

Weitere regelbasierte Sprachen

Neben den genannten regelbasierten Prozessmodellierungssprachen existieren weitere regelbasierte Ansätze, welche jedoch aufgrund gewisser Eigenschaften nur bedingt zur integrierten Modellierung und Ausführung geeignet sind. Sie werden daher im folgenden Abschnitt nur kurz beschrieben.

Generalised Process Structure Grammars (GPSG)

GPSG zur regelbasierten Modellierung

Generalised Process Structure Grammars (GPSG) [52] ist einer der ersten Ansätze zur regelbasierten Modellierung von Geschäftsprozessen. Die Besonderheit an diesem Konzept ist, dass GPSG-Regeln sowohl die Durchführung von Aktivitäten als auch den Zugriff auf Datenobjekte einschränken können. Die organisatorische Perspektive wird jedoch nicht behandelt und es wird nicht zwischen verpflichtenden und empfohlenen Aktionen unterschieden. Des Weiteren bleibt unklar, inwieweit das Konzept implementiert werden kann. Die GPSG können also nicht direkt zur Unterstützung von agilen, personenbezogenen Prozessen eingesetzt werden. Dennoch gilt der Ansatz als grundlage für aktuelle Konzepte.

ESProNa

Die *Engine for Semantic Process Navigation (ESProNa)* wurde von 2009 bis 2011 am Lehrstuhl für Angewandte Informatik IV der Universität Bayreuth entwickelt [66]. Es handelt sich um ein Ausführungssystem für regelbasierte Prozessmodelle. Das System ist in der Logiksprache Prolog implementiert und Prozessmodelle müssen ebenfalls in Prolog verfasst werden. Das System unterstützt zwei Arten von Anfragen: Zum einen lässt sich prüfen, ob ein gegebener Zustand den Regeln des Prozessmodells entspricht, was sich zur Ausführung von Prozessmodellen nutzen lässt. Zum anderen lässt sich der günstigste Pfad von einem Ausgangs- zu einem Zielzustand ermitteln. Dieser Teil des Systems wird als Navigation bezeichnet.

Basierend auf Prolog

Die Navigation ist aufgrund der kombinatorischen Explosion allerdings ohne Weiteres nicht praktikabel. Das verwendete heuristische Suchverfahren erfordert eine feingranulare Gewichtung der Regeln, also umfangreiche Informationen und Annahmen zum Prozess. Die Regelsprache von ESProNa ist nicht definiert oder eingegrenzt. Es handelt sich vielmehr um eine Programmbibliothek, mit der sich regelbasierte Prozesse entwickeln lassen. Der Sprachumfang ist der von Prolog, wodurch sich Prozessmodelle nicht validieren sondern lediglich auf korrekte Prolog-Syntax prüfen lassen. ESProNa stellt selbst auch keine Workflow-Engine dar, sondern deckt lediglich die Regelauswertung ab. Es fehlt zum Beispiel die Verwaltung von Prozessinstanzen, Regeln für die Terminierung von Instanzen und die Schachtelung von Prozessmodellen. Auch wenn das System wertvolle Ideen und Beiträge vor allem in Bezug auf perspektivenübergreifende Modellierung liefert, kann ESProNa also nicht zur adäquaten Modellierung und Ausführung regelbasierter Prozesse eingesetzt werden.

Sprache nicht eingegrenzt

Geschäftsregeln (Business Rules)

Fehlende verhaltensorientierte Perspektive

Geschäftsregeln (engl. business rules) sollen das Verhalten und die Informationen einer Organisation beeinflussen oder führen [145]. Sie lassen dabei die verhaltensbezogene Perspektive naturgemäß außen vor [53], spezifizieren also nur die Randbedingungen, die eingehalten werden sollen und nicht wie genau ein Ziel erreicht werden kann. Geschäftsregeln und Prozesse sind in der Praxis derzeit strikt getrennte Welten, die nur an bestimmten Punkten miteinander verknüpft sind. Der übliche Weg besteht in der Auslagerung komplexer Entscheidungen in Geschäftsregeln.

Business Rule Task

BPMN sieht hierfür die Geschäftsregelaufgabe (*engl. business rule task*) vor. Dieser Aufgabentyp kapselt eine komplexe Entscheidung und damit den Aufruf eines Geschäftsregelmanagementsystems (*Business Rule Management Systems, kurz BRMS)*. Das Ergebnis der Entscheidung liefert die Grundlage für eine nachfolgende Verzweigung. Der Ausgang der Entscheidung steuert den weiteren Verlauf des Prozesses. Das bedeutendste Rahmenwerk zur Abbildung von Geschäftsregeln ist SBVR [98]. Die Semantics of Business Vocabulary and Business Rules (SBVR) ist ein durch die OMG standardisiertes Rahmenwerk für Geschäftsregeln [98]. SBVR erlaubt die Spezifikation von Zusammenhängen und Regeln in einer an Englisch angelehnten natürlichen Sprache. Eine Alternative dazu bietet die im Februar 2014 vorgestellte *Decision Model and Notation (DMN)* [95]. Mit DMN lassen sich Entscheidungen mittels graphischen Diagrammen modellieren. Geschäftsregeln bilden derzeit also keine vollständigen Prozesse ab, sondern kapseln lediglich gewisse logische Bestandteile prozeduraler Prozesse. Obwohl hier also das prozedurale und das regelbasierte Prinzip kombiniert werden, kann die Kombination nicht zur integrierten Unterstützung agiler Prozesse verwendet werden.

SBVR-Regeln

Decision Model and Notation

EM-BrA^{2}CE

Erweiterung von SBVR

EM-BrA^{2}CE macht den ersten Schritt zur Vereinigung von Geschäftsregeln und Prozessen. EM-BrA^{2}CE erweitert die SBVR um Konzepte wie Aktivitäten, Zustände und Bearbeiter und ermöglicht es damit, Prozesse in SBVR auszudrücken [53]. Die Schwierigkeit liegt allerdings in der tatsächlichen Durchsetzung (*engl. enforcement*) dieser Regeln, also der Ausführung der Prozesse. Zur Ausführung müssen die SBVR-Regeln mittels Schablonen in ECA-Regeln (event condition action) überführt werden. Die Schablone

ECA-Regeln

```
<Activity2> may only happen after <Activity1>
```

wird somit in die folgende ECA-Regel übersetzt [44]:

```
on start(<Activity2>): if not completed(<Actitvity1>) then report violation
```

Da nur die in einer Schablone erfassten Geschäftsregeln übersetzt und ausgeführt werden können, geht der ursprüngliche Vorteil durch den Umfang von SBVR verloren. Die Prozesse werden also effektiv mit ECA-Regeln modelliert. Außerdem wird nicht gezeigt, wie mit verschiedenen Modalitäten in den SBVR-Regeln bei deren Übersetzung umgegangen wird. EM-BrA^{2}CE kann also nicht

direkt zur IT-Unterstützung von agilen Prozessen verwendet werden. Die Idee, Geschäftsregeln und Prozesse zu regelbasierten Prozessmodellen zusammenzuführen, wird jedoch im Rahmen dieser Arbeit aufgegriffen.

Die folgenden SBVR Regeln stellen den Beispielprozess aus Abschnitt 1.2.3 in der EM-BrA^{2}CE-Notation dar:

An approve application activity1 can only start after a apply for trip activity2 has completed.
An agent that *has role* administration can perform approve application activity.
It is necessary that an agent that *has been assigned* to an apply for trip activity performs a book flight activity.
An agent that *has role* student can only perform an book flight activity1 after a approve application activity2 has completed.

3.2.5 Zusammenfassung der Evaluation

Nach Betrachtung und Diskussion bestehender, ausführbarer Prozessmodellierungssprachen erfolgt nun eine Zusammenfassung der Evaluationsergebnisse. *Prozedurale* Sprachen können durch zusätzliche Ausdruckssprachen wie beispielsweise *RAL* so erweitert werden, dass komplexe organisatorische Zuweisungsregeln abgebildet werden können. Aufgrund des flussorientierten Modellierungsprinzips, in dem jeder mögliche Ablauf *explizit* abgebildet werden muss, werden prozedurale Modelle bei agilen Prozesse jedoch schnell schwerbis unlesbar.

Regelbasierte Prozessmodellierungssprachen eigenen sich besser zur Abbildung von Agilität. Anstatt jeden möglichen Pfad explizit zu modellieren, wird der Prozess durch eine Menge von Einschränkungen bzw. Regeln auf den Prozessmodellelementen und auf Ereignissen definiert. Das regelbasierte bzw. deklarative Modellierungsprinzip wird vom Declare Rahmenwerk, mit Sprachen wie ConDec und DecSerFlow, und den Sprachen CMMN und DCR-Graphen umgesetzt. Declare ermöglicht außerdem die Unterscheidung von verpflichtenden und optionalen Regeln, wodurch empfohlene Aktionen abgebildet werden können und dem Nutzer zusätzliche Unterstützung gegeben werden kann. Allen ausführbaren regelbasierten Sprachen fehlt jedoch eine ausreichende Unterstützung für komplexere organisatorische Muster. Es ist weder möglich komplexe Zuweisungsregeln abzubilden, noch können perspektivenübergreifende Zusammenhänge, d.h. beispielsweise ein durch die organisatorische Perspektive beeinflusster Kontrollfluss, formuliert werden.

Zusammenfassender Überblick

Tabelle 4 gibt einen abschließenden Überblick über die evaluierten Prozessmodellierungssprachen. Es zeigt sich, dass keine der bestehenden Sprachen die notwendigen Anforderungen erfüllen kann.

Tabelle 4: Evaluation von Modellierungssprachen

Anforderung	Prozedural mit Erweiterungen	Declare	CMMN	DCR-Graph
(A1) Abbildung von Agilität	⊘	✓	✓	✓
(A2) Modalitäten	⊘	✓	⊘	⊘
(A3) Komplexe Zuweisungen	✓	⊘	⊘	⊘
(A4) Perspektivenübergreifend	⊘	⊘	⊘	⊘

3.3 DECLARATIVE PROCESS INTERMEDIATE LANGUAGE (DPIL)

Declarative Process Intermediate Language (DPIL)

Aufgrund der Schwächen bestehender Sprachen wurde, ausgehend von den Anforderungen an die Modellierung agiler, personenbezogener Prozesse, in den letzten Jahren die ***Declarative Process Intermediate Language (DPIL)*** [122], [141], [143] am Lehrstuhl für Angewandte Informatik IV der Universität Bayreuth entwickelt.

Erfüllung der Anforderung

DPIL ist eine textuelle, regelbasierte Prozessmodellierungssprache, mit der Agilität verständlich abgebildet werden kann (**A1**). Die Sprache ist multi-modal, d.h. es können sowohl verpflichtende als auch lediglich empfohlene Regeln definiert werden (**A2**). Neben der verhaltensorientierten Perspektive erlaubt DPIL die Modellierung von organisatorischen Zusammenhängen. Um auch die Abbildung komplexer organisatorischer Zusammenhänge zu ermöglichen, basiert DPIL auf einem flexiblen, generischen Organisations-Metamodell (**A3**). Da eine Regel in DPIL mehrere Perspektiven gleichzeitig betreffen kann, ist es möglich perspektivenübergreifende Zusammenhänge zu modellieren (**A4**). Auf diese Weise erfüllt die Sprache alle genannten Anforderungen und wird daher als Zielsprache für das automatisierte Modellierungsverfahren dieser Arbeit verwendet. Die Ausdrucksstärke von DPIL und die Eignung zur Modellierung von Prozessen wurden bezüglich häufig wiederkehrender Muster in Prozessen (*engl. Workflow Patterns)* evaluiert [143].

3.3.1 Beschreibung der Sprache

Da DPIL als Zielsprache für das Process Mining-Verfahren dieser Arbeit verwendet wird, werden in den folgenden Abschnitten die **Elemente der Sprache** (Strukturelle Modellelemente, Ereignistypen, Regeln), die **konkrete, textuelle Syntax** sowie das **Ausführungsprinzip** von DPIL-Modellen detailliert erläutert. Für eine exakte Beschreibung des DPIL-Metamodells verweisen wir auf [141].

Strukturelle Modellelemente

Strukturelle Elemente von DPIL

Eine Aktivität (*Activity*) stellt in DPIL einen Arbeitsschritt bzw. eine Aufgabe dar. Ein Prozess (*Process*) ist eine zusammengesetzte Aktivität, eine die also wiederum Aktivitäten enthalten kann. Alle weiteren Arten von Aktivitäten sind elementar. Der entsprechende Arbeitsschritt soll also nicht weiter detail-

liert werden. Eine personelle Aufgabe (*Task*) wird während des Prozesses einem oder mehreren menschlichen Prozessteilnehmern zugewiesen. Eine automatische Operation (*Operation*) ist eine Aufgabe, die durch das System ohne Beteiligung von Menschen erledigt wird. Hierbei kann es sich um den Aufruf eines lokalen Programms oder eines externen Dienstes handeln. Ein Verweis auf einen fremden Prozess (*ProcessReference*) ermöglicht die Einbettung von Subprozessen, die unter Umständen in anderen Modellierungssprachen modelliert sein können. Die genannten Elemente decken die funktionale Perspektive der Prozessmodellierung ab.

Ereignisse und Lebenszyklen der Modellelemente

Der entscheidende Bestandteil von DPIL-Prozessmodellen sind Regeln, welche die *Ereignisse* einschränken, die im Lebenszyklus von Aktivitäten auftreten können. Zusätzlich zu den strukturellen Modellelementen sind daher auch die Ereignisse Teil der Sprachdefinition. Alle Ereignisse, die sich auf einen Prozessteilnehmer beziehen oder von einem ausgelöst werden, sind personelle Ereignisse (*IdentityEvent*) und verweisen auf eine durchführende Person (*Performer*). Personelle Aufgaben können begonnen (*Start*) und abgeschlossen (*Complete*) werden. Nur diese beiden Ereignisse werden in den Ereignisverlauf aufgenommen und können daher in Prozessregeln verwendet werden. Operationen, die von Diensten zur Verfügung gestellt werden, können aufgerufen werden (*Invoke*) und kehren nach Abschluss zurück (*Return*).

Ereignisse als Teil der Sprache

Formulierung von Regeln

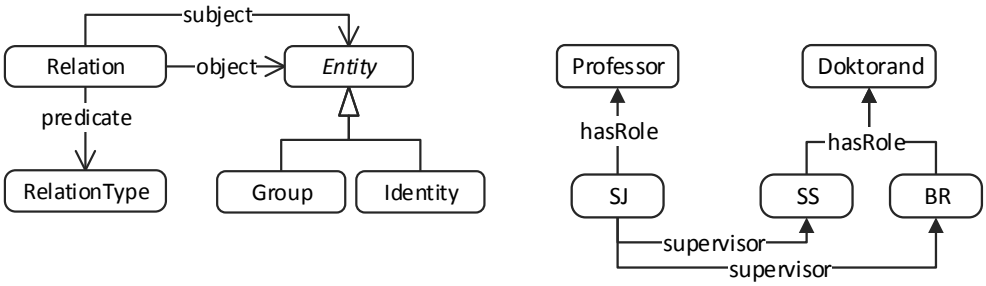
Prozessregeln

Für einen Prozess können Regeln (*ProcessRule*) angegeben werden, welche für die im Prozess definierten Aktivitäten gelten. Dabei kann es sich neben verpflichtenden und empfohlenen Regeln auch um Meilensteine handeln. Um bestimmte Typen von Regelausdrücken benennen und wieder verwenden zu können, können in einem Modell Makros (*Macro*) definiert werden. Regeln und Makros enthalten immer einen Regelausdruck (*Expression*). Ein Ausdruck kann binär (Binary) sein, also eine Konjunktion (*and*), Disjunktion (*or*) oder Implikation (*implies*). Er kann aber auch unär (*unary*) sein, also eine Negation (*not*), der Existenzquantor (*exists*) oder der Allquantor (*forall*). Außerdem kann ein Ausdruck ein Verweis auf ein Prädikat (*PredicateReference*) oder ein Objektauswähler (*ObjectSelector*) sein. Ein Objektauswähler bezieht sich immer auf einen bestimmten Element- oder Ereignistyp (*Type*). Er kann entweder ein Objekt direkt referenzieren (*ObjectReference*) oder mehrere anhand ihrer Eigenschaften eingrenzen (*ObjectPattern*). Objektmuster (*ObjectPattern*) dienen dazu, Objekte anhand ihrer Eigenschaften auszuwählen. Dies ist notwendig, um zum Beispiel die Start-Ereignisse einer Aktivität zu selektieren, um sie anschließend einzuschränken oder von Bedingungen abhängig zu machen. Zur Auswahl von Objekten kann das Objektmuster mit einer Einschränkung (*ConstraintExpression*) versehen werden. Dabei handelt es sich um den Zugriff auf eine Objekteigenschaft (*PropertyAccess*) oder die Konjunktion (*ConstraintAnd*) oder Disjunktion

Regelausdrücke

Objektauswähler

Objektmuster



(a) Organisations-Metamodell **(b)** Organisationsmodell

Abb. 21: Organisations-Metamodell und Beispiel-Modell

(*ConstraintOr*) solcher Zugriffe. Der Zugriff auf eine Objekteigenschaft erfolgt über einen Eigenschaftsschlüssel (*PropertyKey*), der als Alias für die Attribute der Ereignisse dient. Ein Zugriff kann entweder den Wert einer Objekteigenschaft an eine Variable binden (*PropertyBinding*) oder ihn einschränken (*PropertyRestriction*). Durch eine Bindung kann der Wert einer Objekteigenschaft in anderen Teilausdrücken wiederum zur Einschränkung verwendet werden. Einschränkungen geschehen durch den Vergleich einer Objekteigenschaft mit einer Zeichenkette (*StringLiteral*), einer Zahl (*NumberLiteral*) oder einem Verweis (*Reference*). Mit den beschriebenen Sprachelementen erfüllt DPIL die Ausdrucksstärke der Prädikatenlogik 1. Stufe (*engl. first-order logic, kurz FOL*).

Organisations-Metamodell

Generisches, organisatorisches Metamodell

Um reale organisatorische Beziehungen in DPIL-Prozessmodellen abbilden zu können, basiert DPIL auf einem generischen Organisations-Metamodell nach Bussler [30]. Das Metamodell ist in Abbildung 21a dargestellt und besteht aus den folgenden Elementen: *Identity* stellt organisatorische Entitäten dar, welche direkt Aktivitäten zugewiesen werden können, d.h. konkrete Ressourcen bzw. Personen. Der Typ *Group* bezeichnet abstrakte Entitäten, wodurch Mengen von Identitäten beschrieben werden, z.B. Rollen oder organisatorische Gruppen. Der Typ *Relation* stellt die verschiedenen Beziehungen eines bestimmten Beziehungstyps *RelationType* zwischen diesen Elementen dar.

Identitäten

Gruppen

Beziehungen und Beziehungstypen

Aufgrund der generischen Struktur können auf Basis des Metamodells verschiedenste organisatorische Zusammenhänge ausgedrückt werden. Beispielsweise kann definiert werden, dass eine Identität eine bestimmte Rolle hat, dass eine bestimmte Person der bzw. die Vorgesetzte einer anderen Person ist, oder dass eine Person einer bestimmten organisatorischen Einheit angehört. Abbildung 21b zeigt einen Auszug eines exemplarischen Organisationsmodells einer universitären Forschungsgruppe. Es besteht aus zwei Rollen (Professor, Doktorand) und drei Identitäten (SJ, SS, BR), welchen jeweils eine Rolle zugewiesen wird. Außerdem sind mehrere Beziehungen zwischen diesen Identitäten definiert. Diese zeigen beispielsweise an, dass Identität SJ der Vorgesetzte von Identität SS ist.

3.3.2 Konkrete, textuelle Syntax

Konkrete, textuelle DPIL-Syntax

DPIL bietet eine integrierte, textuelle Syntax, mit der sich die strukturellen Elemente und Regeln eines DPIL-Modells definieren lassen. Diese Syntax ist, ausgehend vom DPIL-Metamodell, detailliert in [141] beschrieben. Die abgeleiteten Modelle des Process Mining-Verfahrens dieser Arbeit werden in dieser textuellen Notation abgebildet. Eine graphische Notation für DPIL-Konstrukte ist jedoch für zukünftige Forschungsarbeiten vorgesehen. Zur Darstellung der Grammatik verwendet [141] die Erweiterte Backus-Naur-Form (EBNF) aus der Spezifikation der XML. Jede Regel der Grammatik definiert ein Symbol in der Form

```
Symbol := Ausdruck
```

Modelle

Syntax von Modellen

Ein DPIL-Prozessmodell $M = \{E, R\}$ besteht aus einem strukturellen Teil E und einem Regelteil R. Der strukturelle Teil $E = \{A, I, G, RT\}$ umfasst die Definition der grundlegenden Modellelemente, d.h. die endlichen Mengen der **Aktivitäten** A, der beteiligten **Identitäten** I, der benötigten **Benutzergruppen bzw. Rollen** G sowie der benötigen organisatorischen **Beziehungstypen** RT. Im Kopf des Modells werden die Mengen I, G und RT aus dem Organisationsmodell aufgezählt, damit ihre Namen im Modell verfügbar und referenzierbar sind. Es können hier auch **Makros** definiert werden. Ein Modell kann schließlich einen **Prozess** definieren. Auch ein Modell, das keinen Prozess definiert, ist sinnvoll. Es zählt dann nur Elemente des Organisationsmodells auf oder definiert Makros und kann damit als Bibliothek für andere Modelle dienen.

```
Model:          Identity* Group* RelationType* Macro* Process?
Identity:       'identity' ID
Group:          'group' ID
RelationType:   'relationtype' ID
```

Strukturelle Elemente

Syntax struktureller Elemente

Ein **Prozess** (*process*) kann **Aktivitäten** A und **Prozessregeln** R enthalten. Aktivitäten müssen jeweils eine Kennung (*ID*) besitzen und können daneben einen Namen haben. Eine Aktivität ist eine personelle Aufgabe (*Task*) oder eine Operation (*Operation*). Eine Operation hat keinen Namen sondern nur eine Kennung und einen Rumpf.

```
Process:        'process' ID STRING? '{'
                Activity*
                ProcessRule* '}'
Activity:       Process | Task | Operation
Task:           'task' ID STRING?
Operation:      'operation' ID STRING
```

Regeln

Syntax von Prozessregeln

Die Menge an **Prozessregeln** $R = \{R_H, R_S, R_M\}$ teilt sich auf in verpflichtende Regeln R_H (*ensure*), empfohlene Regeln R_S (*advice*) und Meilensteine R_M (*milestone*). Regeln können eine Kennung und eine Beschreibung besitzen, die vom Rumpf durch einen Doppelpunkt abgetrennt werden müssen. Ein Makro hat eine Kennung und kann über Formalparameter verfügen. Der Rumpf des Makros wird durch *iff* abgetrennt.

```
ProcessRule:        ProcessRuleType (ID? STRING? ':')? Expression
ProcessRuleType:    'ensure' | 'advice' | 'milestone'
Macro:              ID ('(' ID (',' ID)* ')')? 'iff' Expression
```

Die Grammatik realisiert für alle Operatoren eine Infix-Notation, der Operator steht also zwischen den Operanden. Eine Prädikatreferenz ist die Kennung des referenzierten Prädikats (Makro oder Regel) und die Liste der Aktualparameter. Objektauswähler wählen Objekte entweder anhand ihrer Kennung (*ObjectReference*) oder anhand anderer beliebiger Eigenschaften aus (*ObjectConstraint*). Die ausgewählten Objekte können durch einen Doppelpunkt an eine Variable gebunden werden.

```
Expression:           Implies
Implies:              Or ('implies' Or)*
Or:                   And ('or' And)*
And:                  Unary ('and' Unary)*
Unary:                '(' Expression ')'
                      | 'not' Unary
                      | 'exists' Unary
                      | forall
                      | PredicateReference
                      | ObjectSelector
Forall:               'forall' '(' ObjectSelector ObjectSelector+ ')'
PredicateReference:   ID ('(' LiteralOrReference
                      (',' LiteralOrReference)* ')')?
ObjectSelector:       ObjectConstraint | ObjectReference
ObjectReference:      Type ID (':' ID)?
ObjectConstraint:     Type ('(' ConstraintExpression ')')? (':' ID)?
Type:                 'task' | 'operation' | 'process' | 'start' | ...
```

Sollen Objekte nicht anhand ihrer Kennung, sondern anhand von anderen Eigenschaften ausgewählt werden, so ist ein *ObjectConstraint* mit einer entsprechenden *ConstraintExpression* notwendig. Diese hat wieder die Struktur eines Ausdrucks, bildet also einen Baum aus unären und binären Teilausdrücken. Operatoren werden wieder in Infix-Notation realisiert. Die Eigenschaften können entweder an Variablen gebunden werden (*PropertyBinding*) oder mit Variablen oder Konstanten verglichen werden (*PropertyRestriction*).

```
ConstraintExpression:  ConstraintOr
ConstraintOr:          ConstraintAnd ('or' ConstraintAnd)*
ConstraintAnd:         ConstraintUnary ('and' ConstraintUnary)*
ConstraintUnary:       PropertyBinding
                       | PropertyRestriction
                       | '(' ConstraintExpression ')'
PropertyBinding:       PropertyReference ':' ID
PropertyReference:     PropertyKey RQID?
PropertyKey:           'this' | 'of' | 'by' | ...
PropertyRestriction:   PropertyReference Operator? LiteralOrReference
Operator:              '=' | '!=' | ...
LiteralOrReference:    STRING | NUMBER | ID
```

Das Terminal *ID* wird für alle eindeutigen Kennungen verwendet, *RQID* für relative, qualifizierte Bezeichner, *STRING* für konstante Zeichenketten und *NUMBER* für konstante, dezimale Zahlen mit einem Punkt als Dezimaltrennzeichen.

```
ID:         '^'?('a'..'z'|'A'..'Z'|'_') ('a'..'z'|'A'..'Z'|'_'|'0'..'9')*
RQID:       '.' ID ('.' ID)*
STRING:     '"' ( '\\' ('b'|'t'|'n'|'f'|'r'|'u'|'"'|"'"|'\\') |
            !('\\'|'"') )* '"'
NUMBER:     ('0'..'9')+ ('.' ('0'..'9')+)?
```

3.3.3 Agiler, personenbezogener Beispielprozess in DPIL

Beispielprozess in DPIL

Mit der regelbasierten Sprache DPIL kann nun der in Abschnitt 1.2.3 beschriebene, agile und personenbezogene Beispielprozess vollständig und ohne Einschränkungen abgebildet werden. Das Modell in Listing 2 zeigt die Abbildung des Beispiels in DPIL. In den Zeilen 1-6 werden Rollen und Beziehungstypen aus dem Organisationsmodell referenziert und im Modell verfügbar gemacht. In den Zeilen 9-15 werden zur übersichtlicheren Darstellung Makros für häufig auftretende Zusammenhänge definiert, d.h. die Makros *sequence*, *binding*, *role* und *roleSequence*. Beispielsweise entspricht das *sequence*-Makro dem *precedence*-Makro von Declare und besagt, dass Schritt b erst gestartet werden kann, wenn Schritt a irgendwann zuvor beendet wurde. Das *role*-Makro drückt eine rollenbasierte Personenzuweisung aus. Eine Verknüpfung von Zuständigkeiten zwischen zwei Aktivitäten kann durch die Verwendung des *binding*-Makros erfolgen. Das *roleSequence*-Makro stellt schließlich eine perspektivenübergreifende Regel dar, d.h. wenn eine Person mit einer Rolle r Schritt b starten möchte, muss zuvor Schritt a beendet sein.

Die eigentliche Abbildung des Prozesses beginnt in Zeile 18 und startet mit der Definition der Aktivitäten. Die **verhaltensorientierte Perspektive** des Prozesses wird in den Zeilen 24-25 modelliert. Hier wird die verpflichtende zeitliche Abhängigkeit zwischen der Antragsstellung und der Antragsgenehmigung durch eine verpflichtende (*ensure*) *sequence*-Regel abgebildet. Die *advice sequence*-Regel in Zeile 25 drückt die Empfehlung aus, die Antragsgenehmigung vor der

Buchung einer Unterkunft durchzuführen. Die **organisatorische Perspektive** des Prozesses wird in den Zeilen 28-29 modelliert. Durch Verwendung des *role*-Makros wird sichergestellt, dass der Antrag von einem Mitarbeiter der Administration genehmigt wird. Des Weiteren kann durch die *binding*-Regel in Zeile 29 abgebildet werden, dass die Unterkunft vom Antragsteller gebucht werden muss (Verknüpfung von Zuständigkeiten). Aufgrund der lediglich empfohlenen *sequence*-Regel in Zeile 25 ist es zwar nicht empfohlen, eine Unterkunft vor Antragsgenehmigung zu buchen, aber dennoch möglich. Doktoranden müssen sich jedoch an eine feste Reihenfolge dieser Aktivitäten halten. In Zeile 32 wird dieser **perspektivenübergreifende Zusammenhang** durch eine verpflichtende *roleSequence*-Regel abgebildet. Diese besagt, dass im Fall von Doktoranden, der Antrag vor der Buchung einer Unterkunft genehmigt werden muss. In den Zeilen 35-36 werden schließlich die Bedingungen (*milestone*) definiert, die zum Abschluss einer Prozessinstanz erfüllt sein müssen. Hierzu müssen die Aktivitäten "Unterkunft buchen" und "Antrag genehmigen" abgeschlossen sein.

Listing 2: DPIL-Prozessmodell des Beispielprozesses

```
1  %Gruppen bzw. Rollen
2  use group Administration
3  use group Doktorand
4
5  %Beziehungstypen
6  use relationtype hasRole
7
8  %Makros
9  sequence(a,b) iff start(of b at :t) implies complete(of a < t)
10 binding(a,b) iff start(of a by :p) and start(of b) implies start(of b by p)
11 role(t,r) iff start(of t by :p) implies
12               relation(subject p predicate hasRole object r)
13 roleSequence(a,b,r) iff start(of b by :i at :t) and
14                         relation(subject i predicate hasRole object r)
15                         implies complete(of a at < t)
16
17 process Dienstreiseabwicklung {
18     %Funktionale Perspektive
19     task Antragstellen "Antrag stellen"
20     task Antraggenehmigen "Antrag genehmigen"
21     task Unterkunftbuchen "Unterkunft buchen"
22
23     %Verhaltensorientierte Perspektive
24     ensure sequence(Antragstellen,Antraggenehmigen)
25     advice sequence(Antraggenehmigen,Unterkunftbuchen)
26
27     %Organisatorische Perspektive
28     ensure role(Antraggenehmigen, Administration)
29     ensure binding(Antragstellen,Unterkunftbuchen)
30
31     %Perspektivenuebergreifende Regeln
32     ensure roleSequence(Antraggenehmigen,Unterkunftbuchen,Doktorand)
```

```
33
34      %Meilenstein
35      milestone "Done": complete(of Unterkunftbuchen) and
36                        complete(of Antraggenehmigen)
37  }
```

3.3.4 Ausführung von DPIL-Prozessmodellen

Ausführung von DPIL-Modellen

Wird ein Prozess der realen Welt, wie der eben beschriebene Dienstreiseabwicklungsprozess, in einem DPIL-Prozessmodell abgebildet, so lässt er sich durch ein IT-System unterstützen, also „ausführen“. Das Ausführungsprinzip von DPIL-Prozessmodellen wird eingehend in [141], [142] erläutert. Das im folgenden Abschnitt kurz beschriebene Ausführungsprinzip von DPIL-Modellen wird durch das *ProcessNavigation*-System [143] implementiert.

Process Navigation-System

Aktivitäten durchlaufen einen Lebenszyklus aus Ereignissen, der durch das Ausführungssystem verwaltet wird. Eine personelle Aufgabe (*HumanTask*) kann beispielsweise begonnen (*Start*) und abgeschlossen (*Complete*) werden. Der aktuelle Zustand des Prozesses ergibt sich aus dem bisherigen Verlauf an Ereignissen. Jede Aktion im Prozess, wie zum Beispiel das Abschließen einer Aktivität, führt zu einem neuen Ereignis im Verlauf und treibt gleichzeitig die Ausführung des Prozess voran. Hierzu werden zunächst alle möglichen Ereignisse für den nächsten Zeitschritt generiert. Zusammen mit dem bisherigen Verlauf ergeben sich dadurch jeweils mögliche Verläufe für die Prozessinstanz nach dem nächsten Zeitschritt. Diese möglichen Verläufe werden dann auf Basis der im Modell definierten Prozessregeln bewertet. Die Bewertung kategorisiert die möglichen Verläufe (und damit die möglichen nächsten Ereignisse) als nicht erlaubt, nicht empfohlen, neutral oder empfohlen. Jedes Ereignis entspricht einer Aktion im Prozess. Alle nach der Bewertung erlaubten Ereignisse werden entsprechend interpretiert und führen zu den jeweiligen Aktionen im Prozess. Personelle Aufgaben werden zugewiesen oder entzogen und Dienste aufgerufen. Die jeweilige Bewertung einer Aktion als nicht empfohlen, neutral oder empfohlen kann dabei durch die Teilnehmer eingesehen werden. Empfohlene beziehungsweise nicht empfohlene Aktionen können so auf die jeweiligen Prozessregeln zurückgeführt werden.

Ausführungsprinzip von DPIL-Modellen

Bewertung von Ereignissen

3.3.5 Regelfreie Ausführung von Prozessen

In den vorherigen Abschnitten wird beschrieben, wie Prozesse in regelbasierten DPIL-Modellen abgebildet und ausgeführt werden können. Dabei wurde stets davon ausgegangen, dass sämtliche Regeln, die den Prozessverlauf vorgeben, bekannt sind und modelliert werden können.

Häufig nicht alle Regeln bekannt

Besonders zu Beginn der Einführung eines Prozessmanagements fällt es jedoch schwer, sämtliche Regeln vollständig manuell zu definieren, in vielen Fällen erscheint dies sogar unmöglich [83]. Dieses Problem wurde bereits von mehreren Wissenschaftlern identifiziert. De Man schlägt daher die Einführung

Lernphase

einer *Lernphase* vor [83], in der Prozesse in einer Organisation digital aufgezeichnet werden und schließlich durch Sondierung der protokollierten Realität, d.h. durch Anwendung von Process Mining-Techniken, geltende Regeln, d.h. wiederkehrende Muster, identifiziert werden können. Nach einer ausreichend langen Lernphase, in der sämtliche Varianten zumindest einmal aufgetreten sind, entstehen auf diese Weise aussagekräftige Prozessmodelle, welche die Realität des Prozesses beschreiben.

"Flipping the Process Lifecycle"

Swenson schlägt vor, den Process Lifecycle zu drehen (*"Flipping the Process Life Cycle"*) [126]. Anstatt einen Prozess zuerst vollständig zu modellieren und anschließend auszuführen, sollen prozessbeteiligte Mitarbeiter, auf Basis eines IT-basierten Werkzeugs ihre Aufgaben durchführen. Die Tätigkeiten werden auf diese Weise digital protokolliert. Mittels Process Mining-Techniken werden die tatsächlichen Prozesse schließlich sichtbar.

Regelfreie Ausführung

Auf Basis von DPIL und des ProcessNavigation-Systems ist eine Realisierung dieser beiden, lediglich theoretisch und ansatzweise erläuterten Konzepte möglich, genannt *Process Observation*. Dieses Konzept ist bereits in einer eigenen Publikation veröffentlicht [113]. Für die Lern- bzw. Observationsphase der Prozessunterstützung wird ein DPIL-Modell benötigt, dass lediglich einen strukturellen Teil E enthält. Ein Beispiel ist in Listing 3 dargestellt.

Listing 3: Regelfreies DPIL-Prozessmodell des Beispielprozesses

```
%Gruppen bzw. Rollen
use group Professor
use group Doktorand

process Dienstreiseabwicklung {
    %Funktionale Perspektive
    task Antragstellen "Antrag stellen"
    task Antraggenehmigen "Antrag genehmigen"
    task Unterkunftbuchen "Unterkunft buchen"
    ...
}
```

Abbildung von vorhandenem Wissen

In diesem Modell wird lediglich vorhandenes Wissen abgebildet, d.h. Benutzergruppen bzw. Rollen, welche an dem Prozess beteiligt sind, sowie die durchzuführenden Aktivitäten. Je nach Kenntnisstand kann auch mehr oder weniger Information modelliert werden. Hervorzuheben ist, dass keine Prozessregeln R in dem Modell enthalten sind.

Regelbasiertes und regelfreie Ausführung

Wird das obige Modell ausgeführt, d.h. eine neue Prozessinstanz gestartet, ist die Durchführung aller Aktivitäten durch alle definierten Benutzer möglich. Abbildung 22 visualisiert die Benutzeroberfläche beim Starten des Prozesses im regelbasierten und im regelfreien Fall. Betrachten wir zuerst die Oberfläche im regelbasierten Fall. Zur Durchführung der Aktivität "Antrag genehmigen" *muss* beispielsweise zuvor ein Antrag gestellt worden sein. Die Durchführung der Aktivität "Antrag genehmigen" würde daher einer verpflichtenden Regel widersprechen und wird daher vollständig vor dem Nutzer verborgen. Vor Durchführung der Aktivitäten "Flug buchen" und "Unterkunft buchen"

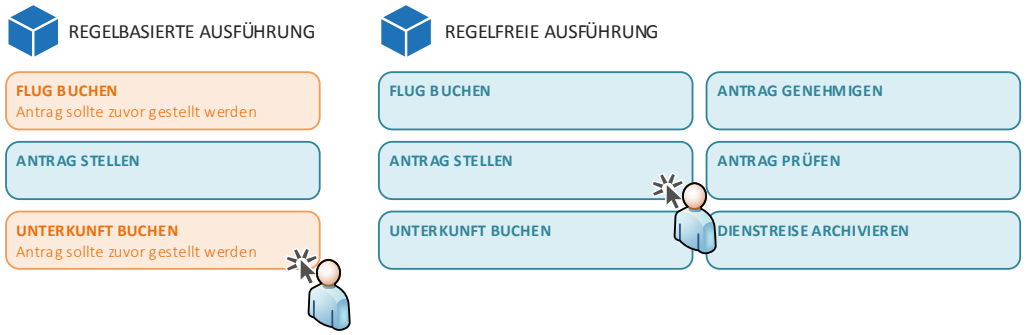


Abb. 22: Regelbasierte und regelfreie Ausführung

sollte ein Antrag gestellt werden. Eine Durchführung ohne Antrag würde zu einem Verstoß einer empfohlenen Regel führen, weshalb diese Schritte als *nicht empfohlen* gekennzeichnet sind. Anders zeigt sich die Benutzeroberfläche im regelfreien Fall. Hier ist zu jedem Zeitpunkt die Durchführung aller Aktivitäten möglich. Während der Ausführung regelfreier Prozessmodelle können, wie in Abschnitt 2.3.5 motiviert, qualitativ hochwertige Ereignisprotokolle erzeugt werden.

3.4 ZUSAMMENFASSUNG

In diesem Kapitel wird die Auswahl und Beschreibung einer passenden **Zielsprache** für das in dieser Arbeit entwickelte Process Mining-Verfahren betrachtet. Auf Basis der Anforderungen an die Modellierung agiler, personenbezogener Prozesse, wurden zuerst existierende Prozessmodellierungssprachen evaluiert.

Das Spektrum an betrachteten existierenden Sprachen teilt sich dabei in **prozedurale Sprachen mit organisatorischen Erweiterungen** und **regelbasierte Sprachen** auf. Prozedurale Sprachen sind, aufgrund mächtiger, organisatorischer Erweiterungssprachen wie *RAL*, im Stande komplexe organisatorische Muster abzubilden. Das prozedurale, flussorientierte Modellierungsprinzip, in dem jeder mögliche Ablauf explizit im Modell abgebildet werden muss, führt im Kontext agiler Prozesse jedoch schnell zu unüberschaubaren Modellen. In regelbasierten Sprachen wie Declare, CMMN oder DCR-Graphen kann diese Agilität hingegen übersichtlich durch Definition einer Menge von Regeln abgebildet werden. In existierenden, regelbasierten Sprachen kann die organisatorische Perspektive von Prozessen jedoch nur rudimentär modelliert werden. Komplexe Zuweisungsregeln, wie die Verknüpfung oder Trennung von Zuständigkeiten sowie Zuweisungsregeln auf Basis organisatorischer Beziehungen, können nicht modelliert werden. Auch perspektivenübergreifende Zusammenhänge, d.h. der Einfluss der organisatorischen Perspektive auf den Prozessablauf, kann nicht abgeleitet werden.

Aufgrund der Schwächen bestehender Sprachen wurde die ***Declarative Process Intermediate Language (DPIL)*** entwickelt. DPIL ist eine textuelle, regelbasierte Prozessmodellierungssprache, mit der Agilität verständlich abgebildet werden kann. Die Sprache ist multi-modal, d.h. es können sowohl verpflichtende als auch lediglich empfohlene Regeln definiert werden. Neben der verhaltensorientierten Perspektive erlaubt DPIL die Modellierung von organisatorischen Zusammenhängen. Um auch die Abbildung komplexer organisatorischer Zusammenhänge zu ermöglichen, basiert DPIL auf einem flexiblen, generischen Organisations-Metamodell. Da eine Regel in DPIL mehrere Perspektiven gleichzeitig betreffen kann, ist es möglich perspektivenübergreifende Zusammenhänge zu modellieren. Auf diese Weise erfüllt die Sprache alle genannten Anforderungen und wird daher als Zielsprache für das automatisierte Modellierungsverfahren dieser Arbeit verwendet.

4 ABLEITEN REGELBASIERTER PROZESSMODELLE AUS LOGS

INHALT

Im vorherigen Abschnitt wurde gezeigt, dass DPIL als regelbasierte Prozessmodellierungssprache gut zur Modellierung agiler, personenbezogener Prozesse geeignet ist. In diesem Kapitel wird ein Verfahren zur automatisierten Ableitung von regelbasierten Prozessmodellen auf Basis der Prozessmodellierungssprache DPIL beschrieben. Die resultierenden Modelle können als Grundlage für weitere Modellierungstätigkeiten, zur Unterstützung der Ausführung oder zur Prozessanalyse verwendet werden.

4.1 ABLEITEN REGELBASIERTER PROZESSMODELLE IN DPIL

Struktureller Teil eines DPIL-Modells

Wie in Kapitel 3 beschrieben wurde, besteht ein DPIL-Prozessmodell $M = \{E, R\}$ aus einem strukturellen Teil E und einem Regelteil R. Der strukturelle Teil

$E = (A, I, G, RT)$ umfasst die Definition der grundlegenden Modellelemente, d.h. die endlichen Mengen $A = \{a_1, a_2, ..., a_n\}$ der Aktivitäten, $I = \{i_1, i_2, ..., i_k\}$ der beteiligten Identitäten, $G = \{g_1, g_2, ..., g_l\}$ der benötigten Benutzergruppen bzw. Rollen sowie $RT = \{rt_1, rt_2, ...rt_o\}$ der benötigen organisatorischen Beziehungstypen. Je nach Art und Komplexität der abzubildenden Zusammenhänge können etwaige Mengen auch leer sein. Müssen beispielsweise nur einfache organisatorische Zusammenhänge abgebildet werden, wie eine direkte Aufgabenzuweisungen zu konkreten Personen, werden keine Gruppen und organisatorische Beziehungstypen benötigt und es gilt $G = \emptyset$ und $RT = \emptyset$.

Regelteil eines DPIL-Modells

Die Menge R definiert eine Menge an Regeln, welche die genannten Modellelemente in Beziehung setzen und darauf basierend die Prozessausführung einschränken bzw. steuern. Die Menge der Regeln $R = \{R_H, R_S, R_M\}$ eines DPIL-Prozessmodells lässt sich in verpflichtende Regeln R_H, Empfehlungen R_S und Meilensteine R_M unterteilen.

Die automatisierte Erzeugung von DPIL-Prozessmodellen muss daher die Generierung des strukturellen Teils sowie des Regelteils des Modells umfassen. Während im strukturellen Teil E die beteiligten Entitäten des Prozesses, d.h. Aktivitäten, Personen, Gruppen und Beziehungstypen definiert werden, werden diese Entitäten im Regelteil wieder aufgegriffen und in Beziehung gesetzt. Der entscheidende Bestandteil von DPIL-Prozessmodellen sind daher Prozessregeln R, welche die Ereignisse einschränken, die unter Berücksichtigung von Personen, Gruppen und Beziehungen im Lebenszyklus von Aktivitäten auftreten können. In der Ableitung der Prozessregeln besteht auch die Komplexität der automatisierten Prozessmodellerzeugung. Wir wenden uns daher zuerst der Erzeugung des Regelteils R von DPIL-Prozessmodellen zu.

4.1.1 Spezifikation und Auswahl von Regelvorlagen

In den folgenden Abschnitten beschäftigen wir uns mit der Erzeugung des Regelteils eines DPIL-Prozessmodells. Regeln stellen die Modellierungselemente einer regelbasierten Prozessmodellierungssprache wie DPIL dar. Theoretisch könnten beliebig komplexe Zusammenhänge zwischen den prozessbeteiligten Entitäten durch Regeln formuliert werden. Sowohl in prozeduralen als auch in anderen regelbasierten Modellierungssprachen genügt jedoch häufig eine relativ kleine Menge an fest definierten Zusammenhängen zur adäquaten Beschreibung von Prozessen. Diese Menge an wiederkehrenden und wichtigen Zusammenhängen wird üblicherweise in einer Modellierungsbibliothek persistiert. Ein häufig verwendetes Element der prozeduralen Prozessmodellierungssprache BPMN ist beispielsweise der Sequenzflusspfeil wodurch die Ablaufreihenfolge zweier Prozessschritte festgelegt wird. Die Semantik des Sequenzflusspfeiles ist fest definiert. Der Modellierer kann lediglich spezifizieren, welche Prozessschritte er mit dem Pfeil verbinden möchte.

Menge an fest definierten Mustern

Regelvorlagen mit typisierten Platzhaltern

In einer regelbasierten Modellierungssprache werden wiederkehrende Zusammenhänge durch sog. **Regelvorlagen** spezifiziert. Im Gegensatz zu konkreten Regeln existieren in Regelvorlagen **freie Platzhalter** in Form von typisierten

Variablen. Durch Angabe von konkreten Werten für diese Parameter durch den Modellierer, wird aus einer Regelvorlage eine konkrete Regel, d.h. die Regelvorlage wird **instanziiert**. Regelvorlagen stellen die Grundlage der automatisierten Erzeugung von regelbasierten Prozessmodellen dar. Durch die Spezifikation bzw. Auswahl einer Menge von Regelvorlagen wird neben der Komplexität der Analyse auch die Ausdrucksstärke des resultierenden Modells festgelegt. Die Auswahl einer bestimmten Menge an Regelvorlagen wird in Kapitel 5.2 erläutert.

Hilfsnotation aus Einleitung

Während in Kap. 1 dieser Arbeit noch eine graphische Hilfsnotation als regelbasierte Sprache ohne definierte Semantik verwendet wird, wird mit DPIL im vorherigen Kapitel eine adäquate Notation identifiziert. Regelvorlagen können nun in der Sprache DPIL formuliert werden.

BEISPIEL (REGELVORLAGE UND KONKRETE REGELN) Ein Beispiel eines häufig benötigten Zusammenhangs ist eine einfache zeitliche Ausführungsreihenfolge zwischen zwei Aktivitäten a_1 und a_2. In der Hilfsnotation der Einführung, wie in Abbildung 2b wird dieser Zusammenhang durch einen Pfeil mit Punkt zwischen Aktivitäten visualisiert. Regeln dieser Form werden in DPIL durch das *sequence*-Makro abgekürzt und sind Instanzen der folgenden DPIL-Regelvorlage:

```
sequence(A1, A2) iff
start(of A2 at :t) implies complete(of A1  at < t)
```

Regelvorlage für verhaltensorientiertes Muster

Dieses Konstrukt besagt, dass ein gewisser Prozessschritt erst gestartet werden kann, nachdem ein bestimmter anderer Prozessschritt beendet wurde, d.h. das Starten (*Start*) eines Schrittes erfordert (*implies*) **zuvor** die Beendigung (*Complete*) eines anderen Schrittes. Die Regelvorlage enthält zwei freie Platzhalter vom Typ *Activity*, im Folgenden abgekürzt durch A. Während der Modellierung eines Prozesses kann diese Regelvorlage mehrfach instanziiert werden, indem für die freien Platzhalter konkrete Aktivitäten eingesetzt werden. Der Modellierer legt dabei nur die konkreten Parameterbelegungen fest, die Definition der Semantik ("A1 **irgendwann vor** A2") ist bereits durch die Regelvorlage gegeben. Möchte ein Modellierer nun beispielsweise abbilden, dass Aktivität a_1 stets beendet sein muss bevor Aktivität a_2 und Activity a_3 gestartet werden können, genügt die Definition von *sequence*(a_1,a_2) und *sequence*(a_1,a_3). Die Regelvorlage des *sequence*-Makros wurde dabei zweifach instanziiert und die folgenden beiden konkreten Regeln definiert:

```
start(of a2 at :t) implies complete(of a1  at < t)
start(of a3 at :t) implies complete(of a1  at < t)
```

Regelvorlagen für organisatorische Muster

Sequence-Regeln setzen zwei Aktivitäten in eine zeitliche Beziehung und beziehen sich rein auf die verhaltensorientierte Perspektive. In agilen, personenbezogenen Prozessen stehen die organisatorische Perspektive sowie perspektivenübergreifende Zusammenhänge jedoch ebenso im Vordergrund. In Kapitel 3 wurde gezeigt, dass sich Regeln in DPIL gleichzeitig auf sämtliche Perspektiven der Prozessmodellierung beziehen können. Neben Regelvorlagen, welche sich rein auf die verhaltensorientierte Perspektive beziehen, können daher in

DPIL auch Regelvorlagen formuliert werden, welche die organisatorische Perspektive einbeziehen.

BEISPIEL (REGELVORLAGE DER ORGANISATORISCHEN PERSPEKTIVE) Ein einfacher, jedoch häufig benötigter Zusammenhang der organisatorischen Perspektive ist beispielsweise die direkte Zuweisung einer Aktivität zu einer bestimmten Person. Dieser Zusammenhang wird durch Instanzen der folgenden Regelvorlage mit dem Makro *direct* abgebildet:

```
direct(A, I) iff
start(of A) implies start(of A by I)
```

Die Regelvorlage enthält wiederum zwei freie Platzhalter, neben einem Parameter vom Typ A nun auch einen Parameter vom Typ *Identity*, kurz I. Möchte der Modellierer nun ausdrücken, dass eine bestimmte Aktivität a_1 stets von einer bestimmten Identität i_1 durchzuführen ist, so instanziiert er diese Regelvorlage mit konkreten Werten und erhält eine Regel der Form

```
start(of a1) implies start(of a1 by i1)
```

Im folgenden Abschnitt bildet eine Menge an Regelvorlagen die Grundlage zur Ableitung des Regelteils von DPIL-Prozessmodellen. Diese stellen *Schablonen* der Zusammenhänge dar, die im gegebenen Ereignisprotokoll entdeckt werden sollen. Die Analyse einer bestimmten Regelvorlage kann als **Suchanfrage** an das gegebene Ereignisprotokoll betrachtet werden. Lösungen der Suchanfrage sind alle Kombinationen von Werten für die Platzhalter, die eine konkrete Regel ergeben, welche im betrachteten Ereignisprotokoll erfüllt ist. Im folgenden Abschnitt wird dieser Suchprozess detailliert beschrieben.

Analyse einer Regelvorlage als Suchanfrage

4.1.2 Generierung und Überprüfung von Regelkandidaten

Das in diesem Abschnitt beschriebene Vorgehen bildet den Kern des in dieser Arbeit beschriebenen regelbasierten Process Mining-Ansatzes und basiert auf den Publikationen [112] und [120] des Autors.

Generierung von Regelkandidaten

Ausgehend von einer Menge an spezifizierten Regelvorlagen können regelbasierte Prozessmodelle automatisiert abgeleitet werden. Im vorherigen Abschnitt werden Regelvorlagen mit freien, typisierten Platzhaltern spezifiziert. Während des **manuellen** Modellierungsvorgangs setzt ein menschlicher Modellierungsexperte auf Basis seines Prozesswissens konkrete Werte für diese freien Variablen ein und formuliert dadurch konkrete Prozessregeln. Im Fall der **automatisierten** Prozessmodellerzeugung ist das Prozesswissen durch ein Ausführungsprotokoll bereits durchgeführter Prozessinstanzen, d.h. Spuren, gegeben. Dieses Wissen wird verwendet, um Prozessregeln automatisiert abzuleiten. Die grundlegende Vorgehensweise ist dabei wie folgt:

Manuelle Modellierung

Automatisierte Modellierung

- **Identifikation** beteiligter Entitäten: Aus den gegebenen Datenquellen (Ereignisprotokoll, Organisationsmodell) werden die prozessbeteiligten Entitäten (Aktivitäten, Identitäten, d.h. beteiligte Personen, Benutzergruppen und Beziehungstypen) extrahiert.

- **Generierung** von Regelkandidaten: Jede gegebene Regelvorlage wird mit **allen möglichen** Kombinationen der identifizierten Entitäten instanziiert. Das Ergebnis ist eine Menge an **Regelkandidaten**.

- **Überprüfung** der erzeugten Regelkandidaten: Die Datenquellen werden ein weiteres Mal untersucht um zu überprüfen, welche Regelkandidaten in den aufgezeichneten Prozessinstanzen gültig und welche verletzt sind.

- **Klassifikation von Regelkandidaten:** Anhand von benutzerdefinierten Schwellenwerten werden Regelkandidaten anschließend klassifiziert und in den Regelteil des Modells übernommen oder verworfen.

Betrachten wir exemplarisch wieder die beiden Regelvorlagen mit den Makros *sequence(A,A)* und *direct(A,I)*. Die Kandidatenerzeugung für diese beiden Regelvorlagen ist in Abbildung 23 visualisiert. Die *sequence*-Regelvorlage besitzt zwei freie Platzhalter vom Typ A. Sei $|A|$ die Anzahl der verschiedenen Aktivitäten, die im betrachteten Ereignisprotokoll auftreten. Dies führt zur Generierung von $|A|^2$ Regelkandidaten der *sequence*-Regelvorlage. Die *direct*-Regelvorlage besitzt hingegen nur einen freien Platzhalter des Typs A und einen des Typs I. Sei $|I|$ die Anzahl der verschiedenen Identitäten die im betrachteten Ereignisprotokoll auftreten. Dies führt zur Generierung von $|A| \cdot |I|$ Regelkandidaten der *direct*-Regelvorlage.

Anzahl an Regelkandidaten der sequence-Vorlage

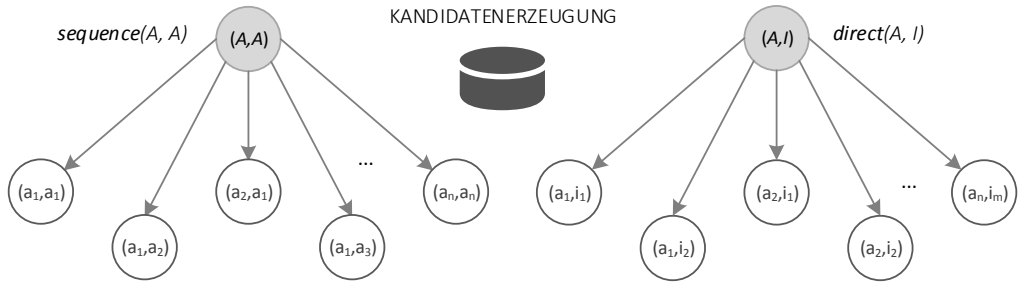


Abb. 23: Regelkandidaten aus exemplarischen Regelvorlagen

Sei $|\Theta|$ die Menge an verschiedenen gegebenen Regelvorlagen und $|P_j(i)|$ die Anzahl der verschiedenen Elemente eines beliebigen, in der Regelvorlage Θ_i vorhandenen, Parametertyps $P_j(i)$ im betrachteten Ereignisprotokoll bzw. Organisationsmodell. Sei $k(i)$ die Anzahl der freien Parameter, d.h. Platzhalter, in Θ_i. Die allgemeine Anzahl an generierten Regelkandidaten $|R_{Cand}|$ er-

Allgemeine Anzahl generierten Regelkandidaten

gibt sich zu $|P_1(1)| \cdot |P_2(1)| \cdot ... \cdot |P_{k(1)}(1)| + |P_1(2)| \cdot |P_2(2)| \cdot ... \cdot |P_{k(2)}(2)| + ... + |P_1(i)| \cdot |P_2(i)| \cdot ... \cdot |P_{k(i)}(i)|$ und daher allgemein zu

$$|R_{Cand}| = \sum_{i=1}^{|\Theta|} (\prod_{j=1}^{k(i)} |P_j(i)|) \tag{1}$$

Überprüfung von Regelkandidaten

Ziel ist es nun herauszufinden, welche der generierten Regelkandidaten tatsächlich während der Prozessausführung erfüllt waren, d.h. welche der Kandidaten tatsächlich gültige Regeln darstellen. Dazu müssen die aufgezeichneten Prozessinstanzen des Ereignisprotokolls überprüft werden. Im folgenden Abschnitt wird das Vorgehen zur Überprüfung von zwei exemplarischen Regeln in jeweils einer Prozessinstanz, d.h. anhand einer Spur, beschrieben.

Überprüfung von Regeln

Betrachtet wird zuerst der Regelkandidat *start(of* a_2 *at* : t*) implies complete(of* a_1 *at* $< t$*)*, d.h. eine Instanz der *sequence*-Regelvorlage. Wird in einer aufgezeichneten Prozessinstanz, d.h. einer speziellen Spur des Ereignisprotokolls, die Aufgabe a_1 beendet bevor die Aufgabe a_2 gestartet wird, so ist diese Regel in der betrachteten Prozessinstanz erfüllt. Die Regel fordert den Abschluss von a_1 vor dem Beginn von a_2 und nutzt dazu eine entsprechende Implikation. Dieser logische Zusammenhang lässt sich, wie in Tabelle 5 gezeigt, als Wahrheitstabelle darstellen.

Logische Implikation

Aus der Definition der logischen Implikation $A \rightarrow B \equiv \neg A \vee B \equiv \neg(A \wedge \neg B)$ ergibt sich, dass die Regel also nur dann nicht erfüllt ist, wenn in der betrachteten Prozessinstanz a_2 begonnen wurde, ohne dass zuvor a_1 abgeschlossen wurde (Zeile 2 in Tabelle 5).
In der Spur

$$\sigma_1: \{\mathbf{s}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_2), \mathbf{c}(a_1)\}$$

ist die Regel somit verletzt, wohingegen sie in der Spur

$$\sigma_2: \{\mathbf{s}(a_1), \mathbf{c}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_2)\}$$

erfüllt ist. Betrachten wir nun die Regel *start(of* a_1*) implies start(of* a_1 *by* i_1*)*, d.h. eine Instanz der *direct*-Regelvorlage. Wird in der betrachteten Prozessin-

Tabelle 5: Wahrheitstabelle zur *sequence*-Regel

start(of a_2 at :t)	**complete(of a_1 at <t)**	**start(of a_2 at :t) $\rightarrow$ complete(of a_1 at <t)**
wahr	wahr	wahr
wahr	falsch	falsch
falsch	wahr	wahr
falsch	falsch	wahr

stanz die Aufgabe a_1 stets von einer Resource i_1 durchgeführt, so ist diese Regel erfüllt. Auch dieser logische Zusammenhang lässt sich wieder in einer Wahrheitstabelle Tabelle 6 darstellen. Die Regel ist also nur dann nicht erfüllt, wenn in der betrachteten Prozessinstanz a_1 nicht von i_1 durchgeführt wird.

In der Spur

$$\sigma_3\colon \{\mathbf{s}(a_1, i_2), \mathbf{s}(a_2, i_2)\}$$

ist die Regel somit verletzt, wohingegen sie in der Spur

$$\sigma_4\colon \{\mathbf{s}(a_1, i_1), \mathbf{s}(a_2, i_1)\}$$

erfüllt ist. Auf diese Weise lässt sich für alle generierten Kandidaten überprüfen, ob eine Regel in einer Prozessinstanz erfüllt oder verletzt ist. Die Gültigkeit einer Regel innerhalb einer *einzigen* Prozessinstanz ist jedoch verständlicherweise wenig aussagekräftig. Ob eine Regel tatsächlich für den gesamten analysierten Prozess gilt, lässt sich nur ableiten, indem sämtliche aufgezeichnete Spuren, d.h. alle Prozessinstanzen, analysiert werden. Das Ergebnis dieser Analyse ist die Anzahl an Prozessinstanzen, welche eine bestimmte Regel erfüllen. Wir bezeichnen diesen Wert als $\mathcal{A}(\mathtt{Rule})$.

Anzahl an Prozessinstanzen, die eine Regel erfüllen

Überprüfen von Bedingungen zur Filterung gehaltloser Regeln

Betrachtet man die beiden Wahrheitstabellen 5 und 6 so zeigt sich, dass eine Regel auch **trivial erfüllt** sein kann. Regeln, die trivial erfüllt sind, werden auch als gehaltlos bzw. als ausdruckslos (*engl. vacous*) bezeichnet. Um aussagekräftige Resultate sicherzustellen, muss während der Analyse unterschieden werden, ob eine Regel nicht-trivial oder trivial erfüllt ist. Betrachten wir dazu den Aufbau einer Regel. Eine DPIL-Regel $A \rightarrow B$ besteht aus zwei Komponenten, einer Bedingung (*engl. condition*) A und einer Konsequenz (*engl. consequence*) B. Eine Regel $A \rightarrow B$ ist nun trivial erfüllt, wenn die Bedingung A bereits **nicht** erfüllt ist. Dieser logische Zusammenhang spiegelt sich z.B. in den letzten beiden Zeilen von Tabelle 5 wieder. In allen Fällen in denen die Bedingung *start(of* a_2 *at* : t*)* (Spalte 1) nicht erfüllt ist, ist die Regel (Spalte 3) trivial erfüllt. In der Spur σ_1 sind beispielsweise alle Regelkandidaten trivial erfüllt, deren Parameter verschieden zu a_1 und a_2 sind.

Trivial erfüllte Regeln

Es ist offensichtlich, dass bei der Überprüfung einer Regel $A \rightarrow B$ nur diejenigen Prozessinstanzen aussagekräftig sind, in denen eine Regel nicht-trivial er-

Tabelle 6: Wahrheitstabelle zur *direct*-Regel

start(of a_1**)**	**start(of** a_1 **by** i_1**)**	**start(of** a_1**)** $\rightarrow$ **start(of** a_1 **by** i_1**)**
wahr	wahr	wahr
wahr	falsch	falsch
falsch	falsch	wahr

füllt ist, d.h. in denen die Bedingung A erfüllt ist. Bei der Überprüfung der Gültigkeit eines Regelkandidaten muss daher zusätzlich überprüft werden, ob die Bedingung des jeweiligen Regelkandidaten erfüllt ist. Das Ergebnis dieser Analyse ist die Anzahl an Prozessinstanzen, in denen eine Regel **nicht-trivial erfüllt** ist. In den folgenden Abschnitten bezeichnen wir diesen Wert als $\mathcal{A}_{NT}(Rule)$.

Nicht-trivial erfüllte Regeln

4.1.3 Support- und Confidence-Rahmenwerk

Als Ergebnis der Analysen des vorherigen Abschnittes steht nun für jeden Regelkandidaten der spezifizierten Regelvorlagen, d.h. der ausgewählten Modellierungselemente, die Anzahl der Prozessinstanzen $\mathcal{A}_{NT}(Rule)$ zur Verfügung, welche die jeweilige Regel nicht-trivial erfüllen. Auf Basis dieser Anzahl können Regelkandidaten nun bewertet und klassifiziert werden. Ziel ist es zu analysieren, ob ein Regelkandidat einen **interessanten**, d.h. häufig vorkommenden, Zusammenhang darstellt und mit welcher **Sicherheit** dieser Zusammenhang gilt. Auf Basis dieser beiden Werte werden bestimmte Regelkandidaten als interessante und gültige Regeln in den Regelteil des resultierenden Modells übernommen und andere als ungültig bzw. tendenziell uninteressant verworfen. Des Weiteren kann darauf basierend eine Klassifikation in verpflichtende und lediglich empfohlene Regeln erfolgen. Zur Bewertung und Klassifikation von Regelkandidaten greifen wir auf zwei Metriken zurück, die auch in klassischen Data Mining-Bereichen Anwendung finden.

Bewertung und Klassifikation von Regelkandidaten

Support

Zur Beurteilung, ob es sich bei einem Regelkandidaten überhaupt um einen interessanten Zusammenhang handelt, wird der sog. **Support**-Wert berechnet. Der Support-Wert $Supp(Rule)$ eines Regelkandidaten ist definiert durch das Verhältnis der Anzahl an Prozessinstanzen, welche die Regel nicht-trivial erfüllen, zur Gesamtanzahl an protokollierten Prozessinstanzen. Der Support-Wert gibt daher Aufschluss darüber, ob es sich bei dem betrachteten Regelkandidaten überhaupt um eine **bemerkenswerte Regel** handelt. Sei $|\Phi|$ die Anzahl an Spuren, d.h. die Anzahl an aufgezeichneten Prozessinstanzen im Ereignisprotokoll Φ. Der Support-Wert eines Regelkandidaten $Rule$ ist definiert durch:

Filterung interessanter Regeln durch Support-Wert

$$Supp(Rule) = \frac{\mathcal{A}_{NT}(Rule)}{|\Phi|} \tag{2}$$

Durch einen Vergleich mit dem benutzerdefinierten Schwellenwert $minSupp$ können tendenziell uninteressante Regelkandidaten verworfen werden. Die Funktionsweise und Notwendigkeit der Berechnung des Support-Wertes eines Regelkandidaten wird anhand eines Beispiels erläutert.

BEISPIEL (BEWERTUNG VON REGELN DURCH DEN SUPPORT-WERT) Der Schwellenwert für interessante Zusammenhänge sei gegeben durch $minSupp = 0.05$,

d.h. eine Regel wird von Analysten nur dann als interessant eingestuft, wenn sie in mindestens 5% der analysierten Prozessinstanzen nachgewiesen wird. Gegeben sei ein Ereignisprotokoll Φ mit $|\Phi| = 1000$, d.h. 1000 protokollierten Prozessinstanzen des zu analysierenden Prozesses. Betrachtet wird der Regelkandidat *sequence*(a_1,a_2), d.h. die Regel, dass eine Aufgabe a_1 beendet sein muss, bevor eine Aufgabe a_2 begonnen werden kann. Die Analyse dieser Regel liefert $\mathcal{A}_{NT}(\textit{sequence}(a_1,a_2)) = 5$, d.h. diese Regel ist in nur 5 Prozessinstanzen nicht-trivial erfüllt. Der Support-Wert für diesen Regelkandidaten ist $\mathrm{Supp}(\textit{sequence}(a_1,a_2)) = 0.005 < \mathrm{minSupp}$. Selbst im Falle dessen, dass a_1 stets beendet wurde bevor a_2 begonnen wurde, handelt es sich bei diesem Regelkandidaten daher um einen eher uninteressanten Zusammenhang und wird daher nicht in den Regelteil des resultierenden Modells übernommen.

Confidence

Klassifikation von Regeln durch Conficence-Wert

Zur Beurteilung ob eine Regel tatsächlich für den analysierten Prozess gilt wird der sog. **Confidence**-Wert berechnet. Der Confidence-Wert $\mathrm{Conf}(\mathrm{Rule})$ eines Regelkandidaten ist definiert durch das Verhältnis des Support-Wertes des Regelkandidaten zu dem Support-Wert der Bedingung des Regelkandidaten. Im Fall von Regelkandidaten, die keine Implikation darstellen, ist die Bedingung in jeder Prozessinstanz erfüllt, daher $\mathrm{Supp}(\mathrm{Condition}) = 1$ und der Confidence-Wert einer Regel entspricht direkt dem Support-Wert. Dieser Wert gibt daher Aufschluss darüber, mit welcher Sicherheit eine Regel tatsächlich gilt. Der Confidence-Wert eines Regelkandidaten Rule ist definiert durch:

$$\mathrm{Conf}(\mathrm{Rule}) = \frac{\mathrm{Supp}(\mathrm{Rule})}{\mathrm{Supp}(\mathrm{Condition})} \tag{3}$$

BEISPIEL (BERECHNUNG DES CONFIDENCE-WERTES) Gegeben sei ein exemplarisches Ereignisprotokoll Φ mit $|\Phi| = 5$. Betrachtet wird wieder die Analyse des Regelkandidaten *sequence*(a_1,a_2). Die Bedingung A der Regel $A \rightarrow B$ ist gegeben durch *start(of* a_2 *at* : t*)*. Tabelle 7 gibt an, in welchen Spuren jeweils die Bedingung und die Regel nicht-trivial erfüllt bzw. verletzt sind.

Tabelle 7: Exemplarisches Ereignisprotokoll und erfüllte Bedingung bzw. Regel

Spur	**Bedingung** A	**Regel** $A \rightarrow B$
$\{\mathbf{s}(a_1), \mathbf{c}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_2)\}$	wahr	wahr
$\{\mathbf{s}(a_1), \mathbf{c}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_2)\}$	wahr	wahr
$\{\mathbf{s}(a_3), \mathbf{s}(a_1), \mathbf{c}(a_1), \mathbf{c}(a_3)\}$	falsch	falsch
$\{\mathbf{s}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_1), \mathbf{c}(a_2)\}$	wahr	falsch
$\{\mathbf{s}(a_1), \mathbf{c}(a_1), \mathbf{s}(a_2), \mathbf{c}(a_2)\}$	wahr	wahr

Zur Berechnung des Confidence-Wertes werden die Support-Werte der Bedingung des Regelkandidaten und des Regelkandidaten selbst benötigt. Da die Bedingung in 4 von 5 Spuren erfüllt ist, ergibt sich $\mathrm{Supp}(\mathrm{Condition}) = 0.8$. Die

Regel selbst ist nicht-trivial in 3 von 5 Spuren erfüllt, d.h. $Supp(Rule) = 0.6$. Der Confidence-Wert des Regelkandidaten ist daher $Conf(Rule) = \frac{0.6}{0.8} = 0.75$. Wir erhalten daher die Information, dass in 75% aller Instanzen, in denen a_2 begonnen wurde (Bedingung), zuvor a_1 beendet wurde.

4.1.4 Klassifikation von Regelkandidaten

Im folgenden Abschnitt werden Regelkandidaten auf Basis benutzerdefinierter Schwellenwerte klassifiziert. Des Weiteren betrachten wir ein Vorgehensmodell zur Spezifikation der Schwellenwerte.

Klassifikation

Klassifikation von Regeln

Der Confidence-Wert wird nun herangezogen, um zu klassifizieren, ob ein Regelkandidat r eine **verpflichtende Regel** darstellt ($r \in R_H$) , d.h. nahezu immer erfüllt ist, eine **empfohlene Regel** repräsentiert ($r \in R_S$), d.h. tendenziell erfüllt ist, oder eine **nicht gültige Regel** ist ($r \notin R$), d.h. in der Mehrheit der relevanten Prozessinstanzen verletzt ist. Wie in Kapitel 3 beschrieben, stehen zur Unterscheidung von verpflichtenden und empfohlenen Regeln in DPIL die Schlüsselwörter *ensure* und *advice* zur Verfügung. Zur Einordnung in diese Bereiche werden zwei benutzerdefinierte Confidence-Schwellenwerte $minConf_S$ und $minConf_H$ benötigt. Diese sind in Abbildung 24 visualisiert.

Confidence-Schwellenwerte

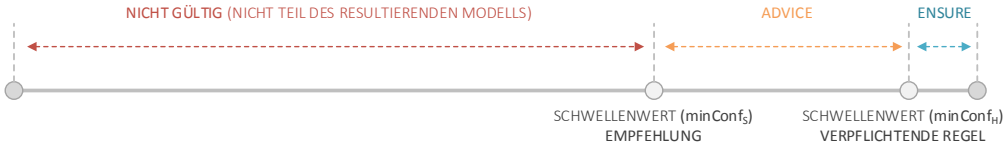


Abb. 24: Klassifikation von Regelkandidaten

Regelkandidaten r, mit $Conf(r) > minConf_H$ werden als verpflichtende Regeln klassifiziert und es gilt $r \in R_H$. Gilt für r hingegen $minConf_S < Conf(r) < minConf_H$, wird der Kandidat als empfohlene Regel klassifiziert und $r \in R_S$. Alle Regelkandidaten mit $Conf(r) < minConf_S$ sind nicht erfüllte Regeln und nicht Teil des resultierenden Prozessmodells, d.h. $r \notin R$. Das folgende Beispiel erläutert die Klassifikation von Regelkandidaten.

BEISPIEL (KLASSIFIKATION VON REGELKANDIDATEN) Gegeben seien Regelkandidaten der *sequence*-Vorlage wie in Abbildung 25 (1) dargestellt, sowie die Schwellenwerte $minConf_S = 0.7$ und $minConf_H = 0.95$ Die Regelüberprüfung liefert die Anzahl der Prozessinstanzen in denen die jeweilige Regel als auch die Bedingung der jeweiligen Regel nicht-trivial erfüllt sind. Auf Basis dieser Anzahl wird für jeden Kandidaten und dessen Bedingung der Support-Wert berechnet. Der Quotient aus diesen Support-Werten führt zu den jeweiligen Confidence-Werten der Regelkandidaten. Abbildung 25 (2) zeigt, dass die Kandidaten *sequence*(a_1,a_2) und *sequence*(a_1,a_3) in allen relevanten Prozessinstan-

zen erfüllt waren. Diese werden daher als verpflichtende Regeln klassifiziert. Die Regel *sequence*(a_2,a_3) hingegen war nur in 80% der relevanten Instanzen erfüllt, überschreitet jedoch den Schwellenschwert $minConf_S$ und wird daher als empfohlene Regel klassifiziert. Die restlichen beiden betrachteten Regelkandidaten sind in nahezu allen relevanten Prozessinstanzen verletzt und werden daher als nicht gültig klassifiziert.

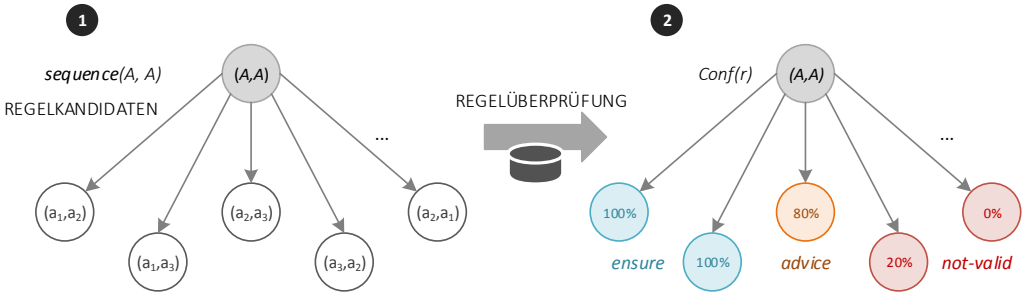


Abb. 25: Exemplarische Klassifikation von Regelkandidaten

Vorgehensmodell zur Spezifikation von Schwellenwerten

Zielkonflikt bei minSupp-Schwellenwert

Zur Ableitung von Modellen müssen demnach die Schwellenwerte minSupp, $minConf_S$, $minConf_H$ vom Benutzer gesetzt werden. Im folgenden Abschnitt wird ein einfaches Vorgehensmodell für die benutzerdefinierten Schwellenwerte aufgezeigt. Widmen wir uns zuerst dem minSupp Schwellenwert. Dieser Wert unterliegt einem Zielkonflikt. Durch einen niedrigen minSupp-Wert steigt die Anzahl an abgeleiteten Regeln. Das abgeleitete Modell wird dadurch unübersichtlicher, enthält jedoch auch seltene, aber potentiell interessante Regeln (minSupp▼ - |R|▲). Wird ein höherer minSupp-Wert gewählt, werden hingehen weniger Regeln abgeleitet. Dadurch, dass das Modell weniger Regeln enthält, wird die Übersichtlichkeit gesteigert. Es werden jedoch seltene, potentiell interessante Zusammenhänge gefiltert (minSupp▲ - |R|▼).

Qualität des Protokolls

Betrachten wir nun den Confidence-Schwellenwert $minConf_H$. Dieser sollte in Abhängigkeit von der Qualität des zu analysierenden Ereignisprotokolls gesetzt werden. Ist beispielsweise bekannt, dass das Protokoll vergleichsweise viele fehlerhafte Aufzeichnungen (*engl. noise*) und Ausnahmefälle (*engl. exceptions*) enthält, so sollte ein niedrigerer $minConf_H$-Wert verwendet werden. Auf diese Weise wird verhindert, dass Fehler und Ausnahmen die Ableitung von verpflichtenden Regeln verfälscht (Qualität Ereignisprotokoll▼ - $minConf_H$▼). Ist bekannt, dass das betrachtete Protokoll kaum Fehler enthält, so kann ein relativ hoher $minConf_H$-Wert verwendet werden (Qualität Ereignisprotokoll▲ - $minConf_H$▲).

Leichte vs. deutliche Tendenzen

Der Confidence-Schwellenwert $minConf_S$ bestimmt die Anzahl an empfohlenen Regeln. Möchte der Nutzer lediglich deutliche Tendenzen als empfohlene Regeln in das Modell übernehmen, so ist ein hoher Schwellenwert zu setzen (Stärke Tendenzen▲ - $minConf_S$▲). Sollen jedoch bereits leichte Tendenzen zur Erfüllung eines Musters als Empfehlung in der Modell übernommen werden,

so bietet sich ein niedriger $minConf_S$-Schwellenwert an (Stärke Tendenzen▼ - $minConf_S$▼).

Inkrementeller Vorgang

Zusammenfassend ist festzuhalten, dass die Wahl der Schwellenwerte sowohl von externen Faktoren (*Qualität von Ereignisprotokollen*) abhängig ist, als auch von individuellen Anforderungen an das resultierende Modell (*Anzahl an Regeln, Stärke von Tendenzen*). Des Weiteren stellt die Wahl der Schwellenwerte in vielen Fällen einen inkrementellen Vorgang dar. Ist die extrahierte Regelmenge im ersten Analysenvorgang beispielsweise noch zu groß, kann der $minSupp$-Schwellenwert Schrittweise so verringert werden, bis sich eine überschaubare Regelmenge ergibt.

4.1.5 Ableiten von Meilensteinen

Meilensteine

DPIL unterstützt ausschließlich die explizite Form der Beendigung einer laufenden Prozessinstanz durch einen sog. **Meilenstein**. Sobald alle definierten Regeln dieses Meilensteins erfüllt sind, gilt eine Prozessinstanz als beendet. Zur vollständigen Erzeugung des Regelteils eines DPIL-Prozessmodells müssen daher auch die Regeln betrachtet werden, die bei Abschluss einer Instanz des betrachteten Prozesses erfüllt sein müssen. Diese sind in der Menge R_M enthalten und werden unabhängig von anderen Regeln betrachtet. Theoretisch sind auch für Meilenstein-Regeln wiederum beliebig komplexe Regelformulierungen möglich. In der Praxis lassen sich jedoch auch hier häufig wiederkehrende Regeltypen, d.h. Regeln einer bestimmten Regelvorlage, identifizieren.

Abschlussbedingung einer Prozessinstanz

Eine typische Abschlussbedingung einer Prozessinstanz ist beispielsweise die Definition von verpflichtend abzuschließenden Aktivitäten. Diese Regeln können durch Instanziierung der Regelvorlage *complete(of A)* abgeleitet werden und dann durch eine Konjunktion verbunden werden.

Des Weiteren können auch bedingte Abschlussbedingungen abgeleitet werden. Ein Beispiel hierfür ist der Zusammenhang, der durch die *response*-Regeln ausgedrückt werden kann. Die Regelvorlage, mit der dieser Typ abgeleitet werden kann, lautet wie folgt:

```
response(A1, A2) iff
complete(of A1 at :t) implies complete(of A2 at > t)
```

Durch diesen Regeltyp wird ausgedrückt, dass vor Abschluss einer Prozessinstanz eine bestimmte Aktivität a_2 durchgeführt werden muss. Dies gilt jedoch nur, wenn zuvor bereits Aktivität a_1 durchgeführt wurde.

Die Überprüfung der Meilenstein-Regeln erfolgt auf die gleiche Art und Weise wie die Überprüfung von Prozessregeln. Lediglich die Unterscheidung in verpflichtende und empfohlene Meilenstein-Regeln entfällt. Jede Abschlussbedingung ist in DPIL generell verpflichtend [141]. Zur Filterung von Fehlern und Ausnahmen wird wie bei Prozessregeln auch der Schwellenwert $minConf_H$ verwendet. Die Ableitung von Meilenstein-Regeln wird anhand eines Beispiels verdeutlicht.

BEISPIEL (ABLEITEN VON MEILENSTEIN-REGELN) Wir betrachten das exemplarische Ereignisprotokoll aus Tabelle 7 und $\mathrm{minConf}_H = 0.95$. Die Abschlussbedingung des betrachteten Prozesses wird anhand der Regelvorlage *complete(of A)* analysiert. Es treten drei verschiedene Aktivitäten in den aufgezeichneten Instanzen auf, was zur Erzeugung von drei Regelkandidaten (*complete(of* a_1*)*, *complete(of* a_2*)*, *complete(of* a_3*)*) führt. Bei allen Kandidaten handelt es sich um Regeln ohne Implikation, weshalb der Confidence-Wert dem Support-Wert entspricht. Es zeigt sich, dass nur *complete(of* a_1*)* in allen Instanzen erfüllt ist ($\mathrm{Conf} = 1$) und Aktivität a_1 offensichtlich stets vor Beendigung des betrachteten Prozesses verpflichtend abzuschließen ist. Die Regel wird daher der Menge R_M des resultierenden Modells hinzugefügt.

4.1.6 Ableiten grundlegender Prozessmodellelemente

Regeln schränken unter Berücksichtigung von Personen, Gruppen und Beziehungen die Durchführung von Aktivitäten ein. Nachdem wir uns in den vorherigen Abschnitten der Erzeugung des Regelteils R des zu analysierenden Prozesses gewidmet haben, betrachten wir nun die Erzeugung des strukturellen Teils $E = (A, I, G, RT)$. Der strukturelle Teil eines DPIL-Prozessmodells umfasst die Definition der grundlegenden Modellelemente, die im Regelteil potentiell verwendet werden, d.h. der Aktivitäten A, der Identitäten I, Benutzergruppen G sowie Beziehungstypen RT. Zur Erzeugung des strukturellen Teils muss daher untersucht werden, welche verschiedenen Modellelemente in dem zu analysierenden Ereignisprotokoll auftreten und welche weiteren Elemente potentiell aus einem zu Grunde liegenden Organisationsmodell zur Beschreibung des Prozesses benötigt werden.

Struktureller Teil eines DPIL-Modells

Betrachten wir zuerst die zu definierenden Mengen an Aktivitäten und Identitäten, d.h. die Mengen A, I. Für jede Aktivität a, für die ein Ereignis in einer der aufgezeichneten Prozessinstanzen $\sigma \in \Phi$ existiert, gilt $a \in A$. Für jede Identität i, die ein Ereignis in einer der aufgezeichneten Prozessinstanzen initiiert hat, gilt $i \in I$. Die Elemente dieser Mengen werden somit unabhängig von der Menge an extrahierten Prozessregeln definiert und können direkt aus dem vorhandenen Ereignisprotokoll abgeleitet werden.

Ableiten von Aktivitäten

Ableiten von Identitäten

Bei der Definition der Elemente der Mengen G und RT bedarf es einer detaillierteren Betrachtung. Die Inhalte dieser Mengen sind unter Umständen abhängig von erzeugten Prozessregeln und können daher erst im Anschluss an die Analyse des Regelteils definiert werden. Wie in Abschnitt 2.3 bereits erläutert wurde, können qualitativ hochwertige Ereignisprotokolle existieren, in denen die Gruppenzugehörigkeit bzw. die Rolle der initiierenden Resource eines Ereignisses bereits im Protokoll vermerkt ist. Diese Gruppen G_{log} können direkt aus dem Protokoll abgeleitet werden. Andere beteiligte Gruppen werden erst nach der Erzeugung des Regelteils sichtbar. Die Regelanalyse kann beispielsweise ergeben, dass eine bestimmte Aktivität stets von Mitgliedern einer bestimmten organisatorischen Gruppe durchgeführt werden sollte. Alle Gruppen, die in mindestens einer abgeleiteten Regel vorkommen, werden in der Menge

Ableiten von Gruppen

Benötigte Beziehungstypen

G_{rules} zusammengefasst. Die resultierende Menge an zu definierenden Gruppen ergibt sich schließlich zu $G = G_{log} \cup G_{rules}$. Ähnlich verhält es sich mit den zu definierenden Beziehungstypen RT. Diese sind im Allgemeinen nicht in Ereignisprotokollen explizit vermerkt, weshalb sich die Menge RT vollständig aus den in abgeleiteten Regeln vorkommenden organisatorischen Beziehungen RT_{rules} ergibt.

4.1.7 Erzeugung regelbasierter DPIL-Prozessmodelle

In den vorherigen Abschnitten wird die Ableitung von grundlegenden Prozessmodellelementen E sowie von Prozessregeln R betrachtet. Anhand dieser Mengen wird nun ein ausführbares DPIL-Prozessmodell erzeugt. Betrachten wir zuerst den Regelteil des Prozessmodells und exemplarisch Regelkandidaten der *sequence*- und *direct*-Regelvorlagen wie in Abbildung 26 visualisiert.

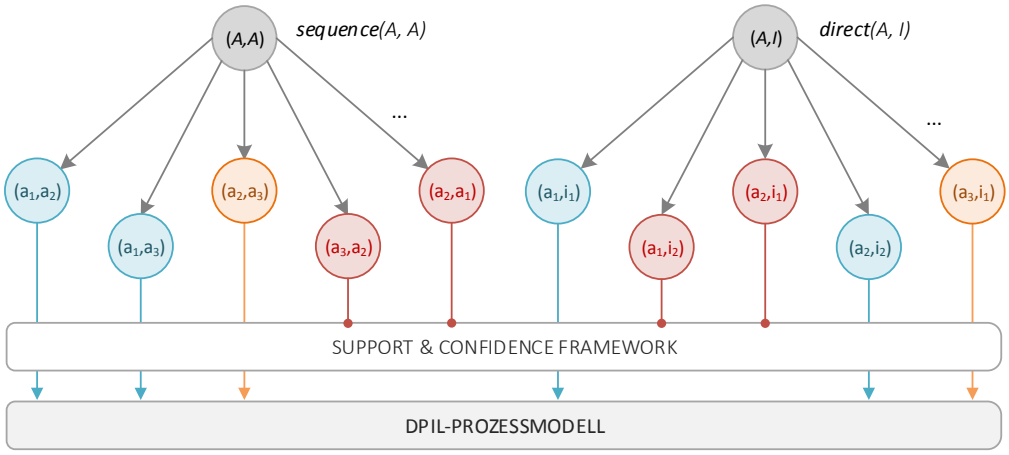


Abb. 26: Erzeugung ausführbarer DPIL-Prozessmodelle

Anhand der Klassifikation durch das Support- und Confidence-Framework werden bestimmte Regelkandidaten als verpflichtende Regeln R_H andere als empfohlene Regeln R_S in das Prozessmodell übernommen. Regelkandidaten r, deren Analyseergebnisse sich ergeben zu $Supp(r) < minSupp$ oder $Conf(r) < minConf_S$, werden nicht in das resultierende Modell übernommen. Das folgende Codeausschnitt 4 zeigt das resultierende DPIL-Prozessmodell.

Listing 4: Erzeugtes DPIL-Prozessmodell des Beispiels

```
1 use identity i1
2 use identity i2
3
4 sequence(a, b) iff
5    start(of b at :t) implies complete(of a at < t)
6
7 direct(a, i) iff
8    start(of a) implies start(of a by i)
9
```

```
10 process Example {
11    task a1
12    task a2
13    task a3
14
15    ensure sequence(a1, a2)
16    ensure sequence(a1, a3)
17    advice sequence(a2, a3)
18
19    ensure direct(a1, i1)
20    ensure direct(a2, i2)
21    advice direct(a3, i1)
22
23    milestone complete(of a1) and complete(of a2) and complete(of a3)
24    ...
25 }
```

In den Zeilen 15-21 sind die konkreten Regelmengen R_H und R_S definiert. Zu beachten ist hier, dass zur einfacheren Lesbarkeit Makros für die entsprechenden Modellierungselemente verwendet werden. Diese parametrisierten Abkürzungen werden zuvor in den Zeilen 4-8 definiert. Der Meilenstein in Zeile 23 spiegelt die abgeleiteten Abschlussbedingungen einer Instanz des Prozesses wider. Demnach müssen zur Beendigung einer Instanz sämtliche definierten Aktivitäten stets mindestens einmal durchgeführt werden. Ausgehend davon, dass das betrachtete Ereignisprotokoll keine expliziten Gruppen enthält und des Weiteren keine Regelvorlagen analysiert werden, die Parameter diesen Typs enthalten, ergibt sich der strukturelle Teil des Modells allein aus den Mengen A und I. In den Zeilen 1-2 sind folglich die beteiligten Identitäten i_1 und i_2 definiert. Die Definition der im betrachteten Ereignisprotokoll auftretenden Aktivitäten a_1, a_2 und a_3 findet in den Zeilen 11-13 statt.

Was wird durch das abgeleitete Modell nun ausgedrückt? Die Analyse des Ereignisprotokolls ergibt, dass eine zeitliche Abhängigkeit zwischen den auftretenden Aktivitäten besteht. Die extrahierten Regeln spiegeln wider, dass Aktivität a_1 offensichtlich verpflichtend **vor** den Aktivitäten a_2 und a_3 durchzuführen ist. Des Weiteren zeigt das Modell, dass a_2 zwar nicht immer vor a_3 durchgeführt wird, jedoch in vergleichsweise vielen Fällen. Es besteht daher eine klare Tendenz, was durch die zwar nicht verpflichtende aber empfohlene Regel in Zeile 17 offensichtlich wird. Die Regeln der Zeilen 19-21 betreffen die organisatorische Perspektive des Prozesses. Die Analyse ergibt hier, dass a_1 und a_2 verpflichtend von den Identitäten i_1 bzw. i_2 durchzuführen sind. a_3 hingehen wird zwar nicht in allen auftretenden Fällen von i_1 durchgeführt, jedoch in verhältnismäßig vielen Fällen. Dies zeigt die empfohlene Regel in Zeile 21. Zur Beendigung einer Instanz muss jede Aktivität mindestens einmal durchgeführt werden (Zeile 23).

4.2 REGELVORLAGEN HÄUFIGER PROZESSMUSTER

Wie in den vorherigen Abschnitten beschrieben, kann der vorgestellte Ansatz ein bestimmtes Prozessmuster aus Ereignisprotokollen rekonstruieren, wenn sich eine parametrisierbare DPIL-Regelvorlage formulieren lässt, welche das Muster abbildet. In diesem Abschnitt definieren wir nun Regelvorlagen und zugehörige Makros, die verwendet werden können, um Modelle von agilen, personenbezogenen Prozessen aus Ereignisprotokollen abzuleiten. Hierzu betrachten wir Regelvorlagen, die Muster der verhaltensorientierten und der organisatorischen Perspektive abbilden. Des Weiteren werden Regelvorlagen beschrieben, die beide Perspektiven kombinieren, d.h. perspektivenübergreifende Zusammenhänge beschreiben.

4.2.1 Verhaltensorientierte Perspektive

Regelvorlagen für häufige verhaltensorientierte Muster

Wir betrachten zuerst Regelvorlagen, die rein die verhaltensorientierte Perspektive betreffen. Die zu Grunde liegenden Muster sind [141] entnommen und basieren auf häufig auftretenden Mustern in Prozessen [4].

- **Sequenz** (*engl. sequence*): Das Sequenz-Muster wurde bereits in einigen Beispielen verwendet und besagt, dass eine gewisse Aktivität erst gestartet werden kann, nachdem eine bestimmte andere Aktivität beendet wurde, d.h. das Starten eines Schrittes erfordert **irgendwann zuvor** die Beendigung eines anderen Schrittes. Zur automatisierten Ableitung dieses Musters wird die folgende Regelvorlage verwendet, die mit dem Makro *sequence* abgekürzt wird. Die Regelvorlage besitzt zwei freie Variablen vom Typ *Activity*, weshalb bei einem Ereignisprotokoll mit $|A|$ verschiedenen Aktivitäten insgesamt $|A|^2$ Regelkandidaten erzeugt werden.

```
sequence(A1, A2) iff
start(of A2 at :t) implies complete(of A1  at < t)
```

- **Direkte Sequenz** (*engl. direct sequence*): Die Regelvorlage mit dem Makro *directSequence* fordert, dass eine gewisse Aktivität nur gestartet werden kann, wenn eine bestimmte andere Aktivität **direkt zuvor** beendet wurde. Eine *directSequence*-Regel schließt eine *sequence*-Regel ein, da jede Aktivität, die direkt vor einer anderen beendet wurde, auch irgendwann vor einer anderen beendet wurde. Auch hier werden $|A|^2$ Regelkandidaten generiert.

```
directSequence(A1, A2) iff
start(of A2 at :ta2) implies complete(of A1 at :ta1) and
not event(at > ta1 and at < ta2)
```

- **Einmalige Durchführung** (*engl. single execution*): Durch die Regelvorlage, die durch das *once*-Makro abgekürzt wird, wird abgeleitet, dass eine

Aktivität nur einmal durchgeführt wird, d.h. der entsprechende Prozessschritt darf nur gestartet werden, wenn er nicht bereits einmal abgeschlossen wurde. Die Vorlage enthält lediglich eine freie Variable, weshalb $|A|$ Kandidaten erzeugt werden.

```
once(A) iff
start(of A at :t) implies not complete(of A at < t)
```

- **Sequentielle Ausführung** (*engl. sequential execution*): Durch die Regelvorlage mit dem Makro *serializeAll* wird abgeleitet, dass sich die Aktivitäten eines Prozesses während der Durchführung nicht zeitlich überschneiden. Da diese Regel für alle Aktivitäten des Prozesses gilt, ist keine Parametrisierung notwendig.

```
serializeAll iff
start(of :a at :t) and not complete(of a at > s) implies
not start(at > t)
```

4.2.2 Organisatorische Perspektive

Da DPIL auf einem flexiblen Organisations-Metamodell basiert, ist es möglich Regelvorlagen zu formulieren, welche den Aufbau von komplexen organisatorischen Beziehungen widerspiegeln. In [111] werden häufig wiederkehrende, organisatorische Muster in Prozessen (*engl. Workflow Resource Patterns, kurz WRPs)* gesammelt, die sich als Evaluierungsrahmenwerk für Prozessmanagementsysteme eignen. Die Muster sind unabhängig von einer spezifischen Implementierung und von den spezifischen Anforderungen einer Domäne. Ursprünglich umfassten die Workflow Patterns ausschließlich die klassischen Perspektiven der Prozessautomatisierung, d.h. Muster der verhaltensorientierten [4] und datenorientierten Perspektive [110] . Die Einbindung menschlicher Prozessteilnehmer, wie bei agilen, personenbezogenen Prozessen, rückte jedoch mehr und mehr in den Vordergrund. Die entsprechende Unterstützung durch Standards der Prozessmodellierung war zunehmend unzureichend. Deshalb wurde das Rahmenwerk der Workflow Patterns um häufig wiederkehrende Muster der organisatorischen Perspektive erweitert [111]. Auf diese Weise lassen sich Modellierungssprachen, Systeme und Process Mining-Ansätze nun auch hinsichtlich ihrer Unterstützung für organisatorische Zusammenhänge sprachunabhängig vergleichen.

Workflow Resource Patterns

Nicht alle WRPs eignen sich zur Evaluation von Modellierungs- bzw. Mining-Fähigkeiten. Einige Muster beziehen sich auch auf die Eigenschaften, die ein Prozessausführungssystem aufweist. Die Gruppe an Mustern die sich am ehesten zur Evaluation von Process Mining-Fähigkeiten eignet, sind die sog. *Creation Patterns*, da sich diese auf Modellierungsmuster beziehen. Diese Gruppe beinhaltet die im folgenden Abschnitt erläuterten Muster, die sich auf die Zuweisung von Personen zu Aktivitäten beziehen. Jedes Muster wird an einem einfachen Beispiel eines universitären Dienstreise-Abwicklungsprozesses erläu-

Creation Patterns

tert. Des Weiteren wird für jedes Muster, soweit möglich, die entsprechende DPIL-Regelvorlage angegeben.

Zuweisungsmuster mit Bezug auf eine Aktivität

Es lassen sich zwei Gruppen an organisatorischen Prozessmustern identifizieren: **Zuweisungsregeln**, die sich auf **genau eine Aktivität** beziehen und Zuweisungsregeln, die sich auf **zwei Aktivitäten** beziehen. Wir betrachten zuerst organisatorische Muster, welche die Zuweisung von Ressourcen betrachten und sich auf genau eine Aktivität beziehen.

- **Direkte Zuweisung** *(engl. Direct Allocation)*: Dieses Muster beschreibt die Zuweisung einer Aktivität zu einer bestimmten, konkreten Identität. Beispielsweise müssen Dienstreiseanträge von der konkreten Person *SJ* genehmigt werden. Dieses Muster wird durch die folgende Regelvorlage mit dem Makro *direct* abgeleitet. Angesichts der freien Platzhalter A und I werden bei einem Ereignisprotokoll mit $|A|$ verschiedenen Aktivitäten und $|I|$ verschiedenen Identitäten $|A| \cdot |I|$ Regelkandidaten erzeugt.

  ```
  direct(A, I) iff
  start(of A) implies start(of A by I)
  ```

- **Rollenbasierte Zuweisung** *(engl. Role-Based Allocation)*: Dieses Muster beschreibt die Zuweisung einer Aktivität zu Personen mit einer bestimmten Rolle. Zur Abwicklung einer Dienstreise müssen gestellte Anträge beispielsweise von Personen mit der Rolle *Sekretariat* auf Richtigkeit überprüft werden. Die zugehörige Regelvorlage wird mit dem Makro *role* bezeichnet. Hier werden Regelkandidaten für jede Kombination aus Aktivität und im Organisationsmodell vorhandener organisatorischer Gruppe erzeugt, d.h. $|A| \cdot |G|$ Regeln.

  ```
  role(A, G) iff
  start(of A by :i) implies relation(subject i predicate hasRole object G)
  ```

- **Fallbehandlung** *(engl. Case Handling)*: Mit diesem Muster wird beschrieben, dass alle Aktivitäten einer bestimmten Prozessinstanz von der gleichen Identität bearbeitet werden müssen. Betrachtet man sämtliche Aktivitäten hinsichtlich der Buchung von Unterkünften und Transportmitteln einer Dienstreise als eigenen Subprozess, so sind alle Aktivitäten von der gleichen Person, d.h. dem Antragsteller, durchzuführen. Die Regelvorlage mit dem Makro *caseHandling* lautet:

  ```
  caseHandling iff
  forall(task A start(of A) implies start(of A by :p))
  ```

- **Fähigkeitsbasierte Zuweisung** *(engl. Capability-Based Allocation)*: Dieses Muster besagt, dass bestimmten Aktivitäten Personen auf Basis derer Fähigkeiten zugewiesen werden. Eine Dienstreiseantrag kann beispielsweise nur von Personen gestellt werden, die mindestens einen Bachelor-Abschluss besitzen. Dieses Muster wird durch die Regelvorlage mit dem Makro *capability* abgeleitet. Betrachtet man die freien Variablen, so werden $|A| \cdot |RT| \cdot |G|$ Regelkandidaten generiert.

```
capability(A, RT, G) iff
start(of A by :p) implies relation(subject p predicate RT object G)
```

- **Organisationsbasierte Zuweisung** *(engl. Organisational Allocation)*: Mit diesem Muster wird beschrieben, dass Aktivitäten Personen auf Basis derer Position in der betrachteten Organisation oder deren Beziehung zu anderen Ressourcen zugewiesen werden. Ein Dienstreiseantrag eines Doktoranden muss beispielsweise von dessen Vorgesetzten genehmigt werden. Für die organisationsbasierte Zuweisung existiert eine Ausprägung, die sich nur auf eine Aktivität bezieht. Dies kann durch die folgende Regelvorlage mit dem Makro *orgDistSingle* abgeleitet werden. Auch hier werden $|A| \cdot |RT| \cdot |G|$ Regelkandidaten erzeugt.

  ```
  orgDistSingle(A, RT, G) iff
  start(of A by :p) implies relation(subject p predicate RT object G)
  ```

- **Verzögerte Zuweisung** *(engl. Deferred Allocation)*: Dieses Muster besagt, dass die Zuweisung einer konkreten Resource zu einer Aufgabe erst zur Laufzeit des Prozesses erfolgt. Da im Allgemeinen kein konkretes Ereignis für die tatsächliche Ressourcenzuweisung in Ereignisprotokollen existiert, ist es nicht möglich zu extrahieren, ob die Zuweisung einer Aktivität zu einer Ressource erst zur Laufzeit des Prozesses erfolgte. Deshalb ist das Muster *Verzögerte Zuweisung* nicht ableitbar.

- **Verlaufsbasierte Zuweisung** *(engl. History-Based Allocation)*: Dieses Muster beschreibt die Zuweisung von Aktivitäten zu Personen auf Basis des bisherigen Verlaufs des Prozesses oder früherer Prozessinstanzen. Ein bestimmter Dienstreiseantrag sollte beispielsweise von einer Person in der Rolle *Sekretariat* überprüft werden, welche bislang am wenigsten Anträge überprüft hat. Da es in DPIL nicht möglich ist Regeln zu formulieren die sich auf andere Prozessinstanzen beziehen, kann das Muster der *verlaufsbasierten Zuweisung* nicht entdeckt werden. In realen Prozessen lassen sich zahlreiche Beispiele dieses Mustern identifizieren, weshalb eine Erkennung dieses Muster durchaus relevant ist und in zukünftigen Forschungsarbeiten berücksichtigt werden sollte. Basis dafür ist jedoch zuerst eine Weiterentwicklung der Sprache DPIL, was in Kapitel 9.3 dieser Arbeit noch einmal aufgegriffen wird.

Zuweisungsmuster mit Bezug auf zwei Aktivitäten

Des Weiteren existieren organisatorische Muster, welche sich auf **zwei Aktivitäten** gleichzeitig beziehen. Muster, die zwischen mehr als zwei Aktivitäten bestehen, können durch mehrere Regeln mit Bezug auf zwei Aktivitäten ausgedrückt werden.

- **Trennung von Zuständigkeiten** *(engl. Separation of Duties)*: Mit diesem Muster wird beschrieben, dass zwei verschiedene Aktivitäten unterschiedlichen Personen zugewiesen werden müssen. Die Genehmigung und die

Überprüfung eines Dienstreiseantrags müssen beispielsweise von unterschiedlichen Personen durchgeführt werden. Zur Entdeckung dieses Musters in Ereignisprotokollen wird die folgende Regelvorlage mit dem Makro *separate* verwendet. Da diese Regelvorlage zwei freie Platzhalter für jeweils eine Aktivität enthält, werden $|A|^2$ Kandidaten erzeugt.

```
separate(A1, A2) iff
start(of A1 by :i) and start(of A2) implies start(of A2 by not i)
```

- **Verknüpfung von Zuständigkeiten** *(engl. Binding of Duties)*: Dieses Muster beschreibt, dass zwei verschiedene Aktivitäten der gleichen Person zugewiesen werden müssen. Beispielsweise müssen Unterkünfte und Transportmittel von der gleichen Person gebucht werden, die auch den Dienstreiseantrag gestellt hat. Zur Entdeckung dieses Musters in Ereignisprotokollen wird die folgende Regelvorlage mit dem Makro *binding* verwendet. Auch hier werden $|A|^2$ Kandidaten generiert.

  ```
  binding(A1, A2) iff
  start(of A1 by :i) and start(of A2) implies start(of A2 by i)
  ```

- **Organisationsbasierte Zuweisung** *(engl. Organisational Allocation)*: Für die organisationsbasierte Zuweisung existiert eine Ausprägung, die sich auf zwei Aktivitäten bezieht. Dies kann durch die folgende Regelvorlage mit dem Makro *orgDistMulti* abgeleitet werden. Zur Analyse dieses organisatorischen Musters müssen $|A|^2 \cdot |\mathsf{RT}|$ Kandidaten erzeugt werden.

  ```
  orgDistMulti(A1, A2, RT) iff
  start(of A1 by :p1) and start(of A2 by :p2) implies
  relation(subject p1 predicate RT object p2)
  ```

Tabelle 8 gibt eine Übersicht über die zuvor beschriebenen organisatorischen Prozessmuster und spezifiziert, soweit möglich, eine entsprechende Regelvorlage zur Ableitung des jeweiligen Musters. Das Pattern *Automatische Ausführung* kann abgeleitet werden, wenn im betrachteten Ereignisprotokoll Ereignisse für die manuelle und automatisierte Aufgabenbearbeitung unterschieden werden. DPIL spezifiziert beispielsweise *invoke*-Ereignisse, um Aufrufe von automatisierten Diensten zu kennzeichnen. Die Tabelle zeigt, dass 8 der 11 organisatorischen Muster mit dem vorgestellten Ansatz entdeckt werden können. Wir setzen dabei voraus, dass für die Ereignisse in dem zu analysierenden Prozessprotokoll jeweils die verantwortliche bzw. initialisierende Ressource (*org:resource* in XES-Dateien) verzeichnet ist. Zur Ableitung der mit einem Stern gekennzeichneten Muster ist zusätzlich ein adäquates Organisationsmodell der betrachteten Domäne notwendig. Alle anderen Muster können direkt aus Ereignisprotokollen ohne weitere Datenquellen abgeleitet werden.

Tabelle 8: Überblick über organisationsbasierte Prozessmuster

	Organisatorisches Muster	DPIL Regelvorlage
Organisatorische Zuweisungsmuster (mit Bezug auf eine Aktivität)		
✓	Direkte Zuweisung	start(of A) implies start(of A by :p)
✓*	Rollenbasierte Zuweisung	start(of A by :p) implies relation(subject p predicate hasRole object G)
⊘	Verzögerte Zuweisung	—
⊘	Authorisierung	—
✓	Fallbehandlung	forall(task A start(of A) implies start(of A by :p)
✓*	Fähigkeitsbasierte Zuweisung	start(of A_1 by :p) implies relation(subject p predicate RT object G)
✓*	Organisationsbasierte Zuweisung	start(of A_1 by :p) implies relation(subject p predicate RT object G)
⊘	Verlaufsbasierte Zuweisung	—
✓	Automatische Ausführung	invoke(A)
Organisatorische Zuweisungsmuster (mit Bezug auf zwei Aktivitäten)		
✓	Trennung von Zuständigkeiten	start(of A_1 by :p) and start(of A_2) implies start(of A_2 by not p)
✓	Verknüpfung von Zuständigkeiten	start(of A_1 by :p) and start(of A_2) implies start(of A_2 by p)
✓*	Organisationsbasierte Zuweisung	start(of A_1 by :p_1) and start(of A_2 by :p_2) implies relation(subject p_1 predicate RT object p_2)

✓* = Organisationsmodell vorausgesetzt

4.2.3 Perspektivenübergreifende Regelvorlagen

Regelvorlagen perspektivenübergreifender Muster

Vor allem in agilen, personenbezogenen Prozessen treten häufig perspektivenübergreifende Zusammenhänge auf (Kapitel 1). Der Ablauf (verhaltensorientierte Perspektive) agiler Prozesse wird vorwiegend bestimmt von menschlichen Akteuren und deren Eigenschaften bzw. deren organisatorischer Position (organisatorische Perspektive). Auch für derartige Zusammenhänge lassen sich entsprechende Regelvorlagen formulieren, auf Basis derer schließlich Regeln abgeleitet werden können. Exemplarisch werden hierzu die beiden Regelvorlagen *roleSequence* und *resourceSequence* definiert, jedoch sind auch beliebige andere Zusammenhänge denkbar.

- **Rollenbasierte Sequenz** (*engl. role-based sequence*): Die rollenbasierte Sequenz besagt, dass eine zeitliche Abhängigkeit zwischen zwei Aktivitäten nur für *Personen in einer bestimmte Rolle* besteht. Aufgrund der freien Parameter werden $|A|^2 \cdot |G|$ Kandidaten erzeugt.

```
roleSequence(A1, A2, G) iff
start(of A2 by :i at :t) and
relation(subject i predicate hasRole object G) implies
complete(of A1 at < t)
```

- **Ressourcenbasierte Sequenz** (*engl. resource-based sequence*): Die ressourcenbasierte Sequenz definiert den gleichen Zusammenhang nicht für eine Rolle sondern für eine konkrete Identität. in Anbetracht der freien Variablen werden hier $|A|^2 \cdot |I|$ Kandidaten erzeugt.

```
resourceSequence(A1, A2, I) iff
start(of A2 by I at :t) implies complete(of A1 at < t)
```

Umgangssprachlich formuliert lassen sich durch die Analyse dieser beiden Regelvorlagen Muster ableiten, die durch das Raster des allgemeingültigeren Sequenz-Musters fallen. Liefert die Analyse der *sequence*-Vorlage keine gültige Regel zwischen den Aktivitäten a_1 und a_2 (da in einigen Instanzen a_1 nach a_2 durchgeführt wurde), kann möglicherweise eine Regel durch zusätzliche Betrachtung der organisatorischen Umstände extrahiert werden. Auf diese Weise wird beispielsweise entdeckt, dass a_1 zwar nicht in allen Fällen vor a_2 durchgeführt wurde, jedoch in allen Fällen in denen a_2 von Personen in der Rolle g_1 bearbeitet wurde.

Adäquate Ausgangsmenge für Analysen

Durch die in diesem Abschnitt aufgezeigte Menge an Regelvorlagen können Muster abgeleitet werden, die von der Forschungsgemeinschaft, in Zusammenarbeit mit Wirtschaft und Industrie, als häufig auftretend identifiziert worden sind. Vor allem in Situationen, in denen Analysten nicht klar ist, nach welchen Zusammenhängen oder Mustern im betrachteten Ereignisprotokoll gesucht werden soll, empfiehlt sich die Verwendung der aufgezeigten Regelvorlagen. Für bestimmte, unter Umständen domänenspezifische, Analyseanforderungen können jedoch auch individuelle Regelvorlagen definiert werden. Dies wird im folgenden Abschnitt beschrieben.

4.3 ERWEITERUNG DES SUCHRAUMS

Die Menge an zu analysierenden Regelvorlagen definiert den Suchraum des Verfahrens und legt gleichzeitig die Ausdrucksmächtigkeit des resultierenden Modells fest. Muster, für die kein Regelkandidat durch Instanziierung einer Regelvorlage erzeugt wird, können auch nicht abgeleitet werden. Die im vorherigen Abschnitt beschriebene Menge an häufig wiederkehrenden Mustern kann durch Instanziierung der zugehörigen Regelvorlagen, im betrachteten Ereignisprotokoll entdeckt werden. Durch Spezifikation weiterer, benutzerdefinierter Regelvorlagen ist das Verfahren stets **erweiterbar**. Auf diese Weise können beispielsweise auch domänenspezifische Zusammenhänge abgeleitet werden. Die Möglichkeit und Notwendigkeit zur Erweiterung des Verfahrens durch benutzerdefinierte Regelvorlagen wird durch das folgende Beispiel verdeutlicht.

Erweiterung durch benutzerdefinierte Regelvorlagen

Rollenbasierte Binding-Regeln

BEISPIEL (ERWEITERUNG DES SUCHRAUMS) Betrachten wir beispielsweise die Regelvorlage mit dem Makro *binding*. Diese besagt, dass zwei Aktivitäten a_1 und a_2 von der gleichen Person durchgeführt werden. Es kann Prozesse geben, in denen zwei Aktvitäten zwar nicht von der gleichen Person, jedoch von

Personen in der gleichen organisatorischen Rolle durchzuführen sind. Dieses Muster kann durch die Analyse der *binding*-Vorlage nicht abgeleitet werden. Zur Entdeckung eines derartigen Zusammenhangs ist es notwendig eine neue Regelvorlage zu spezifizieren, welche im Folgenden als *bindingRole* bezeichnet wird.

```
bindingRole(A1,A2) iff
start(of A1 by :i1) and
relation(subject i1 predicate hasRole object :g) and
start(of A2 by :i2) implies relation(subject i2 predicate hasRole object g)
```

Die Art von Zusammenhängen, die durch diese Regelvorlagen abgeleitet werden können, wird an einem Beispiel erläutert. Gegeben sei die folgende Spur σ_5 und ein Organisationsmodell mit i_1 *hasRole* g_1 und i_2 *hasRole* g_1.

$$\sigma_5\colon \{\mathbf{s}(a_1, i_1), \mathbf{s}(a_2, i_2), \mathbf{c}(a_2, i_2), \mathbf{c}(a_1, i_1)\}$$

Der Regelkandidat *binding*(a_1,a_2) ist in σ_5 verletzt, da a_1 und a_2 von verschiedenen Personen i_1 bzw. i_2 durchgeführt werden. Betrachtet man das gegebene Organisationsmodell, so zeigt sich jedoch, dass beide Identitäten die gleiche organisatorische Rolle g_1 besitzen. Dieser Zusammenhang kann durch die alleinige Analyse der *binding*-Vorlage nicht entdeckt werden. Der Regelkandidat *bindingRole*(a_1,a_2) ist in σ_5 hingegen erfüllt, da die ausführenden Personen von a_1 und a_2 die gleiche organisatorische Rolle besitzen. Durch Erweiterung des Verfahrens durch eine benutzerdefinierte Regelvorlage können somit bislang nicht ableitbare, für Analysten potentiell interessante, Muster entdeckt werden.

4.4 INTEGRIERTE REALISIERUNG VERSCHIEDENER PROCESS MINING-TYPEN

Integrierte Realisierung aller Process Mining-Typen

In Abschnitt 2.2.1 wurden mit *Process Discovery*, *Konformitätsüberprüfung* und *Modell-Verbesserung* die drei Typen von Process Mining eingeführt. Mit dem in den vorherigen Abschnitten beschriebenen Verfahren können alle dieser drei Typen des Process Mining integriert realisiert werden. Dies ist in Abbildung 27 visualisiert und wird im folgenden Abschnitt beschrieben.

Entdeckung (Discovery)

Zur Modell-Entdeckung (*Discovery*) wird nur das zu analysierende Ereignisprotokoll benötigt. Ausgehend von einer benutzerdefinierten Menge an Regelvorlagen, werden Regelkandidaten erzeugt und deren Gültigkeit im Ereignisprotokoll überprüft. Nicht gültige Kandidaten werden verworfen und nur gültige Regeln sind Teil des resultierenden Prozessmodells. Auf diese Weise wird ausgehend von einem Ereignisprotokoll ein DPIL-Prozessmodell erzeugt. Beispiele wurden bereits in den vorherigen Abschnitten detailliert erläutert.

Konformitätsprüfung (Conformance Checking)

Bei der Konformitätsüberprüfung (*Conformance Checking*) wird ein bestehendes DPIL-Modell mit dem vorliegenden Ereignisprotokoll verglichen. In diesem Fall wird eine *gegebene Menge* von Regeln überprüft. Die Erzeugung von

Regelkandidaten entfällt somit. Das Ergebnis der Konformitätsprüfung ist die Information, welche der Regeln des gegebenen DPIL-Modells im betrachteten Ereignisprotokoll erfüllt und welche verletzt sind.

Verbesserung (Enhancement)

Bei der Modell-Verbesserung (*Enhancement*) wird ein bestehendes DPIL-Modell mit dem resultierenden Modell aus einer *Discovery*-Analyse verglichen. Im Grunde beinhaltet die Modell-Verbesserung somit zuerst eine Modell-Entdeckung. Anschließend wird das existierende Modell mit dem abgeleiteten Modell verglichen. Die Fehlerbehebung wird beispielsweise dadurch realisiert, dass bestimmte Regeln aus dem ursprünglichen Modell gelöscht werden, da diese in der Realität, d.h. im betrachteten Ereignisprotokoll nicht gültig sind. Eine Modell-Erweiterung kann beispielsweise dadurch erfolgen, dass zusätzliche Regeltypen, die bislang nicht im Modell enthalten sind, bei der Analyse ausgewählt werden. Enthält das ursprüngliche Modell beispielsweise lediglich Kontrollfluss-Konstrukte (z.B. *sequence*-Regeln), kann das Modell durch Regeln der organisatorischen Perspektive (z.B. *direct*- oder *role*-Regeln) erweitert werden.

Fehlerbehebung

Erweiterung

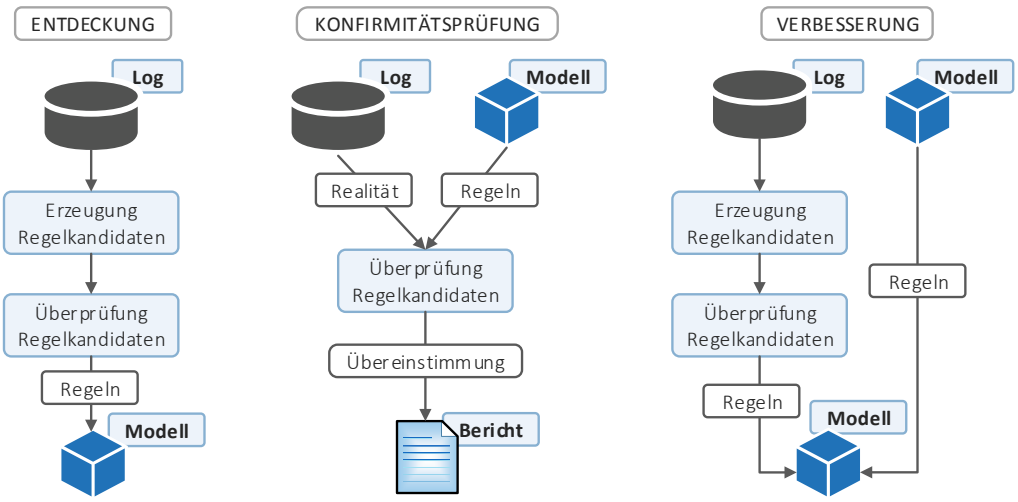


Abb. 27: Integrierte Realisierung der Process Mining-Typen

4.5 IMPLEMENTIERUNG

Regelüberprüfung durch Pattern Matching

In den vorherigen Abschnitten wurde beschrieben, wie DPIL-Prozessmodelle durch Überprüfung der Gültigkeit von Regelkandidaten in den einzelnen Spuren des betrachteten Ereignisprotokoll erzeugt bzw. überprüft werden können. Betrachtet man eine Regel als ein spezielles Muster, dann lässt sich die Regelüberprüfung als Mustererkennungsproblem (*engl. Pattern Matching*) formulieren. Je mehr Regelvorlagen spezifiziert werden, desto mehr Regelkandidaten sind zu überprüfen. Zur Ableitung von Modellen muss daher eine potentiell große Menge an Regelkandidaten möglichst effizient überprüft werden.

Einer naiver Implementierungsansatz wäre nun, jeden Regelkandidaten separat und isoliert von anderen Regelkandidaten für jede Prozessinstanz zu überprüfen. Regelkandidaten verschiedener Regelvorlagen enthalten jedoch häufig identische logische Teilausdrücke. Diese müssen offensichtlich für viele Regelkandidaten nur einmal überprüft werden. Ein Mustererkennungsverfahren, welches das Problem der Erkennung vieler verschiedener Muster mit teilweise identischen Teilausdrücken (*engl. Many Pattern/Many Object Pattern Matching*) besonders effizient implementiert, ist der **Rete-Algorithmus**.

Identische logische Teilausdrücke

BEISPIEL (IDENTISCHE TEILAUSDRÜCKE IN REGELKANDIDATEN) In einem Analysevorgang werden beispielsweise die Regelvorlagen *direct*, *role* und *binding* analysiert. Betrachten wir als Beispiel die Regeln *binding*(a_1,a_2), *direct*(a_2,i_1) und *role*(a_2,g_1). Die konkreten Regeln hierzu lauten wie folgt:

```
start(of a1 by i1) and start(of a2) implies start(of a2 by i1)
start(of a2) implies start(of a2 by i1)
start(of a2 by i1) implies relation(subject i1 predicate hasRole object g1)
```

Betrachtet man diese Regelkandidaten, so zeigt sich, dass selbst Regeln verschiedener Regeltypen häufig gemeinsame Teilausdrücke aufweisen. Alle drei Regeln enthalten beispielsweise den Ausdruck *start(of* a_2 *by* i_1*)*. Dieser Ausdruck tritt in den verschiedenen Regelkandidaten mehrmals auf und muss daher nicht für jede Regel neu evaluiert werden. Diese Tatsache wird durch den Rete-Algorithmus ausgenutzt.

4.5.1 Rete-Algorithmus

Der Rete-Algorithmus ist ein **Mustererkennungsverfahren**, das für eine Menge von Mustern diejenigen bestimmt, die durch eine Menge an Objekten erfüllt werden [51]. Damit eignet sich der Algorithmus zur Überprüfung der Menge an Regelkandidaten (Muster) auf der Basis der Modellelemente und Ereignisse (Objekte) des betrachteten Ereignisprotokolls. Der Rete-Algorithmus erzeugt aus einer Menge von Regeln ein Entscheidungsnetzwerk in Form eines Datenflußgraphen. Der Algorithmus erzeugt aus den zu untersuchenden Mustern ein Netz (*lat. rete*) aus sog. α-Knoten, α-Speichern, β-Knoten und β-Speichern. α-Knoten sind Selektionsbedingungen auf einzelnen Objekten. α-Speicher enthalten Referenzen auf die Objekte, die einem Muster entsprechen. β-Knoten verknüpfen wiederum mehrere einzelne Bedingungen, wobei β-Speicher das Ergebnis dieser Verknüpfung enthalten. Die Faktenbasis bezeichnet man als ***Working Memory***.

Rete-Netzwerk

Faktenbasis (Working Memory)

Als Beispiel soll das Muster $A(x) \wedge B(x,y) \wedge C(y)$ dienen. Die Symbole A, B und C sind logische Prädikate, x und y sind Variablen. Der Rete-Algorithmus baut für die in Abbildung 28 aufgezählten Fakten (1) das entsprechende Rete-Netzwerk auf (2). Für die drei Einzelbedingungen werden drei α-Knoten angelegt und für die Verknüpfungen zwei β-Knoten. Das Ergebnis des letzten

Knotens bildet die Menge der Objekte, die diese Regel erfüllen. Die genaue Vorgehensweise ist implementierungsspezifisch.

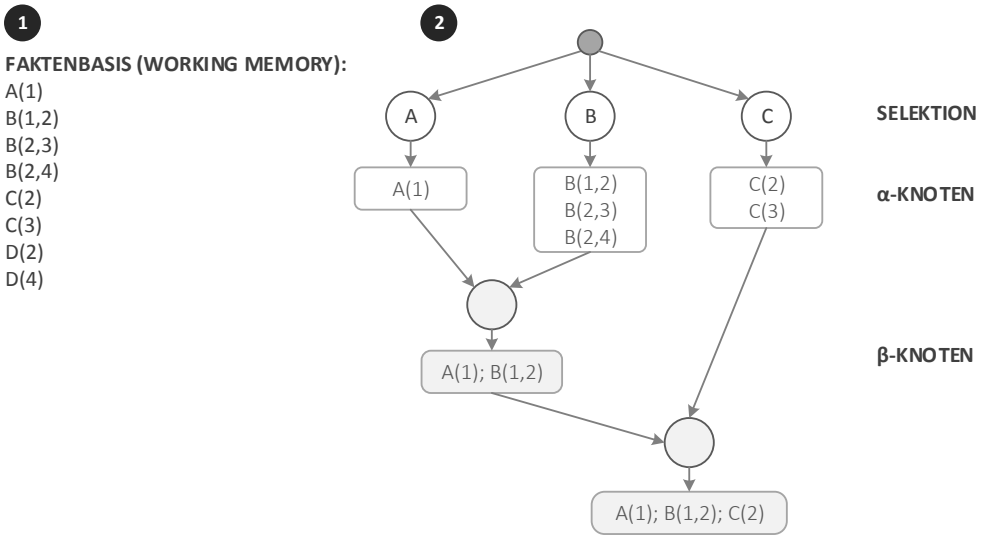


Abb. 28: Exemplarisches Rete-Netzwerk

Vorteil des Verfahrens

Der große Vorteil des Verfahrens liegt in der Überprüfung mehrerer Regeln gleichzeitig. In diesem Fall wird nicht für jede Regel ein eigenes Netz aufgebaut, sondern es werden gleiche Teile der einzelnen Netze verschmolzen. Um alle Regeln auszuwerten, kann dann das resultierende Rete-Netzwerk ausgewertet werden, statt jede Regel einzeln zu prüfen. Der Rete-Algorithmus stellt damit eine effiziente Grundlage für die Überprüfung der Regelkandidaten dar.

4.5.2 Implementierung durch das Drools Framework

Drools-Plattform

Die *Drools-Plattform* [129] beinhaltet eine aktuelle Implementierung des oben gezeigten Rete-Verfahrens. Drools umfasst die Sprache *Drools Rule Language (DRL)*, mit der sich Regeln formulieren lassen. Über entsprechende Schnittstellen lassen sich Objektmengen laden und Regeln auf diesen Objekten auswerten.

Drools Rule Language

Die benötigten Teile des Sprachumfang von DRL werden im Folgenden kurz beschrieben.

Regeln in DRL bestehen aus einer Bedingung (**when**-Teil) und einer Konsequenz (**then**-Teil). Trifft die Bedingung zu, wird die Konsequenz ausgeführt.

DRL-Sprachumfang

Als Bedingung einer Regel unterstützt die DRL den Sprachumfang der Prädikatenlogik erster Stufe (*engl. first order logic*) und ist somit äquivalent zu DPIL. Der grundlegende Aufbau einer DRL Regel ist wie folgt:

```
rule Id
when ...
then ...
end
```

Muster bestehen aus Objekten und Einschränkungen

Bedingungen sind aus Mustern zusammengesetzt. Ein Muster (*engl. pattern*) für Objekte besteht aus dem Typ der Objekte und Einschränkungen (*engl. constraints*) ihrer Eigenschaften. Die zu einem Muster passenden Objekte lassen sich bei jeder Übereinstimmung an eine Variable binden und somit innerhalb der Bedingung wiederverwenden. Folgendes Muster passt zu allen weiblichen Kunden, die älter als 35 Jahre sind, und bindet die entsprechenden Objekte jeweils an die Variable c:

```
$c: Customer(sex == FEMALE, age > 35)
```

Auch die Eigenschaften der Objekte lassen sich an eine Variable binden. So wird im Folgenden das Alter alle männlichen Kunden jeweils an die Variable $a gebunden:

```
Customer(sex == MALE, $a: age)
```

Die logische **Konjunktion** zweier Muster wird mit ***and*** gebildet. Folgender Ausdruck liefert alle Kombinationen aus Kunden und Lieferanten, deren Ort übereinstimmt:

```
Customer($c: city) and Supplier(city == $c)
```

Die logische **Disjunktion** wird mit ***or*** gebildet. Regeln, die eine Disjunktion enthalten werden in eine disjunkte Normalform transformiert, sodass effektiv mehrere unabhängige Regeln entstehen. Aus diesem Grund ist der Geltungsbereich gebundener Variablen unterhalb einer Disjunktion auf den jeweiligen Ast der Disjunktion beschränkt. Die Disjunktion erlaubt dadurch eine wahlweise Bindung von Variablen. So wird im folgenden Beispiel die Variable $pensioner an die Vereinigung der beiden Objektmengen gebunden:

```
$pensioner: (Person(sex == FEMALE, age > 60) or
Person(sex == MALE, age > 65))
```

Der **Existenzquantor** wird mit ***exists*** gebildet. Der entsprechende Ausdruck wird höchstens einmal ausgewertet und es werden keine Variablen gebunden. Folgende Regel fordert, dass mindestens eine Frau existiert, die älter als 35 Jahre ist:

```
exists(Customer(sex == FEMALE, age > 35))
```

Der **Allquantor** wird mit ***forall*** gebildet. Der Ausdruck ist wahr, wenn alle Fakten, die auf das erste Muster passen, auch auf alle weiteren Muster passen. Variablen, die innerhalb des Allquantors gebunden werden, sind außerhalb von ihm nicht sichtbar. Folgende Regel fordert, dass alle englischen Busse rot sind:

```
forall($bus: Bus(type == "english") Bus(this == $bus, color = "red"))
```

Die **Negation** wird schließlich mit ***not*** gebildet:

```
not(Bus(color == "red"))
```

Im folgenden Abschnitt wird nun beschrieben, wie die Drools-Plattform zur Regelüberprüfung verwendet werden kann.

4.5.3 Verwendung von Drools zur DPIL-Regelüberprüfung

Überprüfung von Regeln mit Drools

Die Gültigkeit der Menge an Regelkandidaten wird für jede Spur, d.h. jede Prozessinstanz, separat überprüft. Vor der Überprüfung der aktuell betrachteten Prozessinstanz wird ein Arbeitsspeicher (*Working Memory*) angelegt, der sämtliche Fakten der aktuellen Prozessinstanz enthält, also die grundlegenden Modellelemente aus dem betrachteten Ereignisprotokoll und des Organisationsmodells sowie den Ereignisverlauf, d.h. die Ereignisse der betrachteten Spur des Ereignisprotokolls. Um die Drools-Plattform zur Überprüfung der Regelkandidaten nutzen zu können, müssen diese in der DRL-Sprache vorliegen. Die Transformation von DPIL- zu DRL-Ausdrücken ist ausführlich in [141] beschrieben. Die wichtigsten Teilausdrücke bei der Transformation von DPIL zu DRL sind in Tabelle 9 aufgeführt.

Transformation von DPIL zu DRL

Tabelle 9: Transformationregeln bei der Generierung von DRL aus DPIL

Nr	DPIL Ausdruck	DRL Ausdruck
1	activity :a	$a: Activity()
2	activity A :a	$a: Activity(id == "A")
3	start(**of** A)	$a: Activity(id == "A") **and** Start(Activity == $a)

4	expr	**rule** Id
		when
		  expr
		then
		  listener.onRuleOccured(drools.getRule()));

5	x **implies** y	**not** (x **and not** y)

DPIL-Regeln werden wie in Zeile (4) in DRL-Regeln übersetzt. Dabei wird die gesamte DPIL-Regel in den ***when***-Teil der DRL-Regel transformiert. Die Konsequenz im ***then***-Teil enthält lediglich einen Methodenaufruf, der die Erfüllung der betrachteten Regel an die umliegende Infrarstruktur meldet. Da DRL keine Implikation unterstützt, muss eine Implikation in DPIL wie in Zeile (5) in einen äquivalenten Ausdruck entsprechend der Identität $A \rightarrow B \equiv \neg(A \wedge \neg B)$ in DRL übersetzt werden.

Betrachten wir als Beispiel die *sequence*-Regel

```
start(of a2 at :t) implies complete(of a1 at < t)
```

Die Anwendung der Transformationsregeln aus Tabelle 9 lautet die DRL-Repräsentation zur Überprüfung der vollständigen Regel wie folgt:

```
rule sequence(a1,a2)
when
 a2 : Activity(id == "a2") and
 a1 : Activity(id == "a1") and not
   (Start(Activity == a2, t: time) and not (Complete(Activity == a1, Time < t)))
then
 listener.onRuleOccured(drools.getRule());
end
```

Wie bereits erläutert, muss zur nicht-trivialen Erfüllung einer Regel auch die Bedingung der Regel erfüllt sein. Zur Überprüfung eines Regelkandidaten ist daher die Erzeugung von zwei DRL-Ausdrücken erforderlich. Die DRL-Regel zur Überprüfung der Bedingung lautet:

```
rule condition_sequence(a1,a2)
when
 a2 : Activity(id == "a2") and
 a1 : Activity(id == "a1") and Start(Activity == a2, t: time)
then
 listener.onRuleOccured(drools.getRule());
end
```

Betrachten wir als weiteres Beispiel die *direct*-Regel

```
start(of a1) implies start(of a1 by i1)
```

Die DRL-Repräsentation zur Überprüfung der vollständigen Regel lautet wie folgt:

```
rule direct(a1,i1)
when
 a1 : Activity(id == "a1") and
 i1 : Identity(id == "i1")
 and not (Start(Activity == a1, Performer == i1))
then
 listener.onRuleOccured(drools.getRule());
end
```

Die DRL-Regel zur Überprüfung der Bedingung lautet:

```
rule condition_direct(a1,i1)
when
 a1 : Activity(id == "a1") and
 i1 : Identity(id == "i1") and Start(Activity == a1)
then
 listener.onRuleOccured(drools.getRule());
end
```

In entsprechenden Listen wird für jeden Regelkandidaten die Anzahl an erfüllten Bedingungen bzw. die Anzahl an erfüllten Regeln selbst verwaltet. Darauf basierend können die entsprechenden Support- und Confidence-Werte, wie in Abschnitt 4.1.3 beschrieben, berechnet und tatsächlich gültige Regeln abgeleitet werden.

4.5.4 Parallelisierung der Regelüberprüfung

Regelüberprüfung auf Basis von Instanzen parallelisieren

Ereignisprotokolle realer Anwendungsfälle enthalten häufig die Ereignisinformationen von mehreren tausend Prozessinstanzen. Wie erläutert, muss die Gültigkeit der Regelkandidaten für jede aufgezeichnete Prozessinstanz überprüft werden. Die Überprüfung einer Menge von Regelkandidaten in einer Prozessinstanz ist dabei unabhängig von der Überprüfung in einer anderen Instanz.

Auch die zeitliche Reihenfolge, in der die einzelnen Instanzen überprüft werden, spielt während der Analyse keine Rolle. Dies ermöglicht eine effiziente Parallelisierung der Regelüberprüfung für eine große Menge an Prozessinstanzen (Abbildung 29).

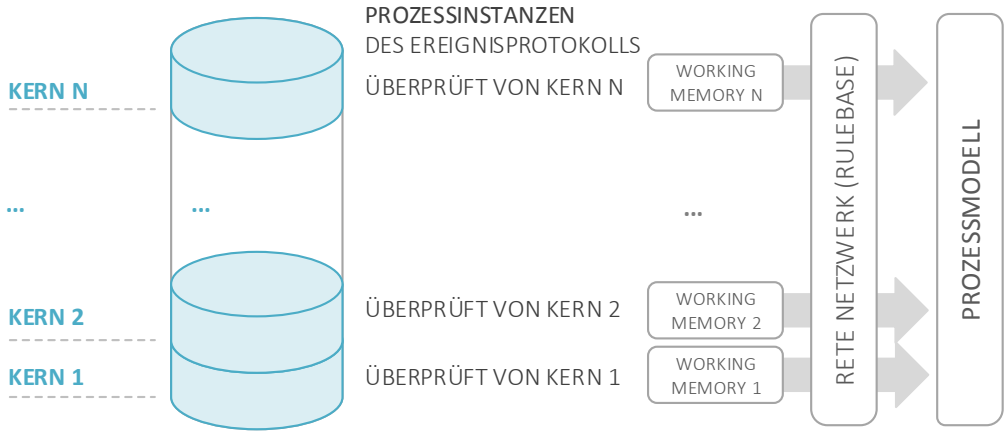


Abb. 29: Parallelisierung der Regelüberprüfung

Einmaliger Aufbau des Rete-Netzwerk

Die Strukturen für den Rete-Algorithmus, d.h. das Rete-Netzwerk, müssen dazu nur einmal aufgebaut werden und können von da an über die gesamte Dauer des Analysevorgangs hinweg wiederverwendet werden. Für jeden Prozessorkern wird ein separates Working Memory definiert, in den die Ereignisinformationen dessen aktuell betrachteter Prozessinstanz geladen werden.

4.5.5 Vergleich mit alternativen Realisierungsmethoden

RDF und SPARQL

Im Laufe dieser Dissertation sind verschiedene Implementierungsvarianten zur Regelüberprüfung evaluiert worden. Neben dem Rete-Algorithmus und Verwendung des Drools-Frameworks kann das Überprüfen der Regelkandidaten auch auf Basis von *SPARQL* [61] und einer *RDF*-Transformation des gegebenen Ereignisprotokolls erfolgen. SPARQL ist eine graphbasierte Anfragesprache für RDF (*Resource Description Framework*) [72] und ein Standard des World Wide Web Consortiums (W3C). Der Name SPARQL ist ein rekursives Akronym für *SPARQL Protocol And RDF Query Language*. Ein RDF-Modell ist ein Datenmodell mit formaler Semantik, das auf gerichteten Graphen basiert. Daten werden in RDF als Aussagen über einzelne Objekte dargestellt. Diese Aussagen werden dabei als Tripel formuliert. Ein Tripel ist eine Elementaraussage, die aus Subjekt, Prädikat und Objekt besteht. Die Menge der Tripel bildet einen Graphen und wird als RDF-Modell bezeichnet.

Ereignisprotokoll als RDF-Modell

RDF Graph-Datenbanken

Es existieren zahlreiche plattformunabhängige Graph-Datenbanken auf RDF-Basis, wie beispielsweise *OpenRDF Sesame* [29] oder *Apache Jena* [36], in denen ein Ereignisprotokoll als RDF-Graph abgebildet werden kann. Für nahezu alle aktuellen Programmiersprachen existieren außerdem entsprechende Programmierbibliotheken, die den Umgang mit RDF und SPARQL kapseln. In

SPARQL Ausdrucksstärke

SPARQL können Anfragen an einen RDF-Graphen formuliert werden, welche die Ausdrucksstärke der Prädikatenlogik 1. Stufe besitzen [18], [99]. Da die Ausdrucksstärke daher identisch zu der von DPIL ist, können somit sämtliche DPIL-Regeln auch als SPARQL-Anfrage ausgedrückt werden. In Listing 5 ist beispielsweise die Überprüfung des Regelkandidaten *sequence*(a_1,a_2) als SPARQL-Anfrage formuliert. Die Anfrage liefert *true*, wenn die Regel **nicht erfüllt** ist, d.h. wenn es ein Start-Ereignis e_1 zur Aktivität a_2 gibt, jedoch kein Complete-Ereignis e_2 zu a_1 zuvor. Wird kein Gegenbeispiel im Graph gefunden, so liefert die Anfrage *false* und die Regel ist in der betrachteten Spur erfüllt.

Sequence-Regel als SPARQL-Anfrage

Listing 5: Sequence-Regel als SPARQL-Anfrage

```
ASK {
        ?e1 a :start.
        ?e1 :activity :a2.
        ?e1 :time ?t1.

        FILTER NOT EXISTS {
                ?e2 a :complete.
                ?e2 :activity :a1.
                ?e2 :time ?t2.  FILTER (?t2 < ?t1)
        }
}
```

Auf ähnliche Weise kann auch die Überprüfung der Regel *direct*(a_1,i_1) als SPARQL Anfrage formuliert werden. Diese Anfrage ist in Listing 6 dargestellt. Diese Anfrage liefert *true*, wenn es ein Start-Ereignis e_1 zur Aktivität a_1 gibt, das nicht von der Identität i_1 initiiert wird. Auf diese Weise nachgewiesen, ob die Regel in der betrachteten Instanz erfüllt oder verletzt ist.

Direct-Regel als SPARQL-Anfrage

Listing 6: Direct-Regel als SPARQL-Anfrage

```
ASK {
        ?e1 a :start.
        ?e1 :activity :a1.

        FILTER NOT EXISTS {
                ?e1 :identity :i1
        }
}
```

Zusammenfassend weißt die Implementierung auf Basis von SPARQL/RDF dabei die folgenden Vor- und Nachteile auf:

⊛ W3C Standard und plattformunabhängige Unterstützung

⊖ Abbildung von Ereignisprotokoll auf RDF-Graph notwendig
⊖ Keine kombinierte Betrachtung mehrerer Regeln
⊖ Komplizierte Transformation von DPIL Regeln zu SPARQL Anfragen

Vorteile der Rete/Drools-Lösung

Die Implementierung auf Basis des Rete-Algorithmus und Drools weißt deutliche Vorteile gegenüber der RDF/SPARQL-Lösung auf. Einerseits ist keine Transformation eines Logs notwendig. Die Methode kann direkt auf das Objektmodell des Ereignisprotokolls angewendet werden. Des Weiteren werden durch den Rete-Algorithmus gemeinsame logische Ausdrücke von Regeln lediglich einmal überprüft. Bei Verwendung von SPARQL stellt die Überprüfung jeder Regel eine separate Anfrage dar. Drools und dessen Regelsprache DRL unterstützen die direkte Abbildung von Prädikatenlogik 1.Stufe, weshalb die Transformation von DPIL zu DRL erheblich leichter und verständlicher ist als die Abbildung von DPIL-Regeln in SPARQL-Anfragen. Ein Nachteil des Rete-Algorithmus ist hingegen die teilweise aufwändige Generierung des Rete-Netzwerks, d.h. das Aufsuchen von gemeinsamen logischen Teilausdrücken in der zu überprüfenden Regelmenge. Eine abschließende Zusammenfassung der Vor- und Nachteile der Rete/Drools-Implementierung ist wie folgt:

⊛ Anwendung direkt auf Objektmodell des Ereignisprotokolls möglich
⊛ Kombinierte Betrachtung von Regeln mit gemeinsamen Teilausdrücken
⊛ Direkte Abbildung von Prädikatenlogik 1. Stufe

⊖ Teilweise rechenintensive Generierung des Rete-Netzwerks

4.6 ZUSAMMENFASSUNG

In diesem Kapitel wurde der grundlegende Ansatz zur Ableitung von regelbasierten Prozessmodellen in der Sprache DPIL erläutert. Die Methode basiert auf sog. **Regelvorlagen**, die den grundlegenden Aufbau von Mustern darstellen, nach denen in einem gegebenen Ereignisprotokoll gesucht werden soll. Regelvorlagen enthalten **freie Platzhalter** eines bestimmten Typs. Durch Einsetzen von allen möglichen Kombinationen für diese Variablen werden **Regelkandidaten** generiert. Die Gültigkeit jeder dieser Regelkandidaten wird anschließend für jede Spur des Ereignisprotokolls überprüft. Auf diese Weise wird aufgedeckt, in welcher Anzahl an Prozessinstanzen Regelkandidaten erfüllt oder verletzt sind. Auf Basis dieser Anzahl und eines **Support/Confidence-Rahmenwerks** werden unwichtige von wichtigen und gültige von ungültigen Regeln getrennt. Gültige und interessante Regeln werden anschließend als **verpflichtend** oder lediglich **empfohlen** klassifiziert und in das resultierende DPIL-Prozessmodell übernommen.

Des Weiteren wurden schließlich Regelvorlagen angegeben, die sich besonders zur Analyse von Ereignisprotokollen agiler, personenbezogener Prozesse eignen. Neben Regelvorlagen zur Ableitung von verhaltensorientierten Zusammenhängen, wurden insbesondere Regelvorlagen definiert, durch deren Analyse **organisationsbasierte Prozessmuster** und **perspektivenübergreifende Muster** abgeleitet werden können. Durch Definition neuer, benutzerdefinierter Regelvorlagen ist das vorliegende Verfahren stets erweiterbar.

Das vorgestellte, regelbasierte Process Mining-Verfahren wurde auf Basis des **Rete-Algorithmus** implementiert. Regelkandidaten verschiedener Regelvorlagen enthalten häufig identische Teilausdrücke, welche für viele Regelkandidaten nur einmal überprüft werden müssen. Diese Eigenschaft wird durch den Rete-Algorithmus ausgenutzt. Die **Drools**-Plattform beinhaltet eine aktuelle Implementierung des Verfahrens und wird daher zur Implementierung der Regelüberprüfung verwendet.

5 KONFIGURATION UND VORVERARBEITUNG

INHALT

In diesem Kapitel wird betrachtet, wie das vorgestellte Verfahren zur automatisierten Erzeugung von regelbasierten Prozessmodellen konfiguriert werden kann und der Suchraum durch eine Vorverarbeitung des betrachteten Ereignisprotokolls sinnvoll eingeschränkt werden kann.

5.1 MOTIVATION

Das bislang vorgestellte Verfahren analysiert **vollständig** das gegebene Prozessereignisprotokoll auf Basis einer spezifizierten Menge von Regelvorlagen. Dabei werden **bestimmte Regelvorlagen** mit **allen möglichen Parameterkombinationen** belegt und auf diese Weise Regelkandidaten generiert, die anschließend für **alle aufgezeichneten** Prozessinstanzen überprüft werden. Betrachtet man diese Vorgehensweise und die hervorgehobenen Wörter, stellen sich folgende Fragen:

Auswahl von Regelvorlagen

- **Welche Regelvorlagen sollen analysiert werden?** Die Auswahl einer spezifischen Menge an Regelvorlagen ermöglicht die Analyse des aufgezeichneten Prozesses aus einer bestimmten Sicht. Resultierende Prozessmodelle sind daher immer nur eine spezielle Sicht auf die Realität. Ob das resultierende Modell nützlich ist oder nicht, wird durch die Fragen bestimmt, die durch die Analyse beantwortet werden sollen. Ein Modell sollte die Dinge herausstellen, die für einen bestimmten Analysten relevant sind.

Dies wird durch die Auswahl der zu analysierenden Regelvorlagen erreicht, wodurch das Verfahren konfiguriert und die Ausdrucksstärke des resultierenden Modells festgelegt wird. Außerdem wird die zu analysierende Menge an Regelvorlagen von der Qualität der Datenbasis beeinflusst. Nach Zusammenhängen, die aufgrund fehlender Information nicht gefolgert werden können, muss gar nicht erst gesucht werden.

Auswahl von Teilprozessen

- **Soll das gesamte Ereignisprotokoll überprüft werden?** Prozesse bestehen in der Realität in vielen Fällen sowohl aus strikten als auch aus agilen Teilprozessen. Der in dieser Arbeit vorgestellte Ansatz ist zur Analyse der agilen Teilprozesse gut geeignet. Betrachtete Ereignisprotokolle enthalten jedoch häufig auch die Ereignisse von strikten Teilprozessen. Um aussagekräftige Prozessmodelle erzeugen zu können, müssen die Ereignisse der agilen Teile des betrachteten Prozesses identifiziert werden. Nur dieser Teil des Ereignisprotokolls wird anschließend mit dem vorliegenden Verfahren analysiert.

Auswahl von Regelkandidaten

- **Macht es Sinn, alle möglichen Regelkandidaten zu generieren?** In realen Ereignisprotokollen tritt eine Vielzahl an verschiedenen Entitäten, d.h. beispielsweise verschiedenen Aktivitäten, auf. Das exemplarische Log eines Krankenhausinformationssystems [28] enthält beispielsweise ca. 600 verschiedene Aktivitäten. Allein die Analyse der *sequence*-Regelvorlage führt in diesem Fall zu rund $600^2 = 360000$ Regelkandidaten, welche alle für jede Prozessinstanz überprüft werden müssen. Viele der verwendeten Parameterkombinationen treten dabei jedoch nie zusammen in einer Prozessinstanz auf. Beispielsweise macht es wenig Sinn, einen Regelkandidaten *sequence*(“Flug buchen”, “Zug buchen”) zu erzeugen und zu überprüfen, da diese Aktivitäten nur in Ausnahmefällen zusammen in einer Prozessinstanz auftreten. Die Erzeugung aller möglichen Kombinationen führt daher zu unnötiger Komplexität. Stattdessen ist eine intelligente Erzeugung von Regelkandidaten zur Abstraktion von seltenen bzw. nicht auftretenden Kombinationen sinnvoll. Dies kann durch eine Vorverarbeitung bzw. Voranalyse des betrachteten Ereignisprotokolls erreicht werden.

In den folgenden Abschnitten werden wir uns der Lösung dieser Probleme widmen. Dies wird durch eine adäquate Konfiguration des Verfahren und durch eine Vorverarbeitung des gegebenen Ereignisprotokolls erreicht.

5.2 VORGEHENSMODELL ZUR WAHL VON REGELVORLAGEN

In diesem Abschnitt wird beschrieben, wie der Anwender eine adäquate Menge an Regelvorlagen zur Analyse eines vorliegenden Ereignisprotokolls wählt bzw. definiert. Die Menge an Regelvorlagen wird dabei einerseits vom Ziel der Analyse beeinflusst, andererseits von der Qualität der vorliegenden Datenbasis.

Beide Einflussfaktoren sind bei der Wahl der zu analysierenden Schablonen zu berücksichtigen.

5.2.1 Auswahl auf Basis der analysierten Perspektiven

Modelle stellen bestimmte Sicht auf Realität dar

Prozesse der Realität können sehr komplex sein und viele verschiedene Dimensionen besitzen. Ein Prozessmodell ermöglicht immer nur eine **bestimmte Sicht** auf diese komplexe Realität [11]. Das Modell sollte die Dinge herausstellen, die für einen bestimmten Analysten relevant sind. Inwieweit ein Modell verwendbare Einblicke liefert, ist davon abhängig, welche Fragen durch die Analyse beantwortet werden sollen. Auch Prozessmodelle, die aus Ereignisdaten gewonnen werden, stellen spezielle Sichten auf diese Realität dar. Eine Sicht sollte eine **zweckmässige Abstraktion** des Verhaltens eines Ereignisprotokolls darstellen. Auf ein Ereignisprotokoll kann es durchaus mehrere nützliche Sichten geben. Verschiedene Analysten benötigen eventuell auch unterschiedliche Sichten. Die erzeugten Modelle können den Fokus dabei auf unterschiedliche Perspektiven legen, beispielsweise auf die organisatorische Perspektive des Prozesses. Im klassischen Process Mining-Bereich erfolgt die Auswahl der gewünschten Sicht durch die Wahl eines geeigneten Process Mining-Algorithmus, mit dem das Ereignisprotokoll analysiert wird. Im Fall des in dieser Arbeit vorgestellten Ansatzes erfolgt die Auswahl der Sicht auf den Prozess durch die Wahl einer bestimmten Menge von Regelvorlagen. In Abbildung 30 sind diese *benutzerdefinierten Sichten* exemplarisch dargestellt. Während in Sicht 1 lediglich die verhaltensorientierte Perspektive betrachtet wird, beispielsweise durch die Analyse von *sequence*-Regeln (schwarzer Pfeil), wird in Sicht 2 die verhaltensorientierte Perspektive unter Berücksichtigung der organisatorischen Perspektive betrachtet, exemplarisch durch die Analyse von *roleSequence*-Regeln (blauer Pfeil mit Akteursymbol). Betrachten wir exemplarisch wieder den Dienstreiseprozess aus Abschnitt 1.2.3. In Sicht 1 wird ersichtlich, dass der Antrag immer gestellt werden muss bevor er genehmigt werden kann. Die Unterkunft kann in bestimmten Fällen auch vor der Antragstellung gebucht werden, auch wenn für Doktoranden eine Antragstellung vor Buchung der Unterkunft verpflichtend ist. Da in Sicht 1 die organisatorische Perspektive nicht einbezogen wird, ist dieser Zusammenhang hier nicht enthalten. In Sicht 2 hingehen wird die Ausführungsreihenfolge der Aktivitäten unter Berücksichtigung der organisatorischen Perspektive dargestellt. Hier enthält das abgeleitete Modell den beschriebenen perspektivenübergreifenden Zusammenhang, dargestellt durch Pfeil mit Akteursymbol.

Zweckmäßige Abstraktion

Benutzerdefinierte Sichten durch Regelvorlagen

Verhaltensorientierte Sicht

Soll beispielsweise rein die **verhaltensorientierte Perspektive**, d.h. der Kontrollfluss, des Ereignisprotokolls analysiert werden, so können sämtliche Regelvorlagen ignoriert werden, welche andere Perspektiven betrachten bzw. einbeziehen. Eine adäquate Menge Θ_1 an Regelvorlagen zur Konfiguration des Verfahrens ist hier beispielsweise

$$\Theta_1 = \{sequence(A_1,A_2), directSequence(A_1,A_2), once(A), serializeAll\}$$

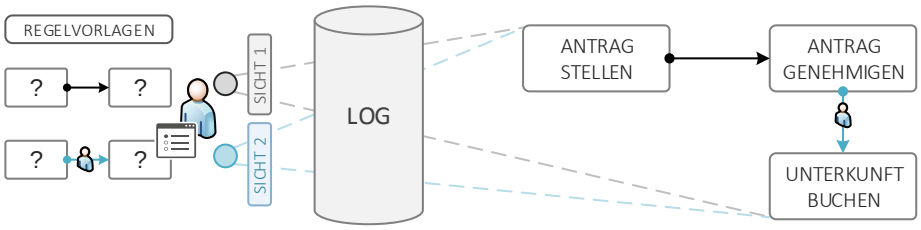


Abb. 30: Benutzerdefinierte Sichten auf den Prozess

Organisationsorientierte Sicht

Ist ein Analyst hingegen an **organisatorischen Zusammenhängen** interessiert, d.h. daran, welche Aktivitäten von welchen Personen durchgeführt werden müssen bzw. dürfen, dann bieten sich die folgenden Mengen Θ_2 und Θ_3 an Regelvorlagen an.

$$\Theta_2 = \{direct(A,I), role(A,G), capability(A,RT,G), orgDistS(A,RT,G)\}$$

Während Θ_2 Zuweisungsregeln ableitet, die genau eine Aktivität betreffen, werden in Θ_3 Zuweisungsregeln abgeleitet, die zwei Aktivitäten betreffen.

$$\Theta_3 = \{binding(A_1,A_2), separate(A_1,A_2), orgDistM(A_1,A_2,RT)\}$$

Perspektivenübergreifende Sicht

Soll der Einfluss von Personen auf den Kontrollfluss des Prozesses untersucht werden, stehen **perspektivenübergreifende Zusammenhänge** im Vordergrund, d.h. der Einfluss der organisatorischen auf die verhaltensorientierte Perspektive. Hierfür bietet es sich an das gegebene Ereignisprotokoll auf Basis der Menge Θ_4 an Regelvorlagen zu analysieren:

$$\Theta_4 = \{sequence(A_1,A_2), roleSequence(A_1,A_2,G), resourceSequence(A_1,A_2,I)\}$$

Durch die Auswahl einer entsprechenden Menge an Regelvorlagen kann der Suchraum eingeschränkt werden. Dadurch wird die Geschwindigkeit der Analyse erhöht und die Verständlichkeit des resultierenden Modells verbessert, da das Modell nur die gewünschte Information enthält.

5.2.2 Auswahl auf Basis der Qualität der Datenbasis

Qualität der Datenbasis

Die Menge an zu analysierenden Regelvorlagen ist auch von der Qualität der vorliegenden Datenbasis abhängig. Nach Zusammenhängen, die aufgrund fehlender Information nicht gefolgert werden können, muss gar nicht erst gesucht werden. In Abschnitt 2.3.3 sind die Grundanforderungen an ein Ereignisprotokoll definiert, die zur Anwendung jeglicher Process Mining-Verfahren erfüllt sein müssen. Abgesehen von diesen Grundanforderungen können Ereignisprotokolle teilweise erhebliche Qualitätsunterschiede aufweisen, d.h. mehr oder weniger Informationen zu Prozessereignissen enthalten. Des Weiteren sind eben-

so Verfügbarkeit und Qualität von Organisationsmodellen der betrachteten Domäne zu berücksichtigen. In den folgenden Abschnitten wird gezeigt, wie sich die unterschiedliche Qualität der Datenbasis auf Definition und Auswahl von Regelvorlagen auswirkt. In Listing 7 ist exemplarisch ein Ereignis eines XES-basierten Ereignisprotokolls abgebildet, das verschiedene Attribute enthält.

Listing 7: Exemplarisches Ereignis eines Ereignisprotokolls

```
<event>
        <string key="org:resource" value="SS"/>
        <string key="org:group" value="Doktorand"/>
        <date key="time:timestamp" value="2013-11-06T14:58:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
```

Ereignistyp-Attribut

Ereignistyp-Attribut in sequence-Vorlage

In Listing 7 ist der Ereignistyp durch das *lifecycle:transition* XES-Attribut abgebildet. In den bisher definierten Regelvorlagen wird der Ereignistyp explizit verwendet. Betrachten wir beispielsweise die *sequence*-Vorlage:

```
start(of A2 at :t) implies complete(of A1 at < t)
```

In dieser Schablone sind Ereignisse vom Typ *start* und *complete* enthalten. In den Grundanforderungen an Protokolle wird nicht spezifiziert, dass zu einem Prozessereignis (*event*) auch der Ereignistyp enthalten sein muss. Es kann daher auch Logs geben, die kein Attribut für den Ereignistyp aufweisen. Liegt ein derartiges Ereignisprotokoll vor, kann bei der Analyse nicht zwischen verschiedenen Ereignistypen unterschieden werden. Die *sequence*-Vorlage muss daher entsprecht angepasst werden:

Angepasste Regelvorlage

```
event(of A2 at :t) implies event(of A1 at < t)
```

Die Regelvorlage ist nun weniger spezifisch, da nicht mehr zwischen verschiedenen Ereignistypen unterschieden wird. Auf Basis dieser umdefinierten *sequence*-Regelvorlage kann nun jedoch auch in Ereignisprotokollen nach zeitlichen Abhängigen gesucht werden, die kein Attribut für den Ereignistyp enthalten. Auf ähnliche Weise müssen auch anderen Regelvorlagen entsprechend dem Informationsgehalt des zu analysierenden Ereignisprotokolls angepasst werden.

Ressourcen-Attribut (RA) und Organisationsmodell (OM)

Ressourcen-Attribute in Ereignisprotokollen

Auch Attribute, welche den initiierenden Akteur oder verantwortliche Gruppe einer Aktivität bzw. eines Ereignissen beschreiben, sind in den Grundanforderungen von Logs nicht gegeben. Je nach Verfügbarkeit von Ressourcen-Attributen im betrachteten Ereignisprotokoll müssen daher verschiedene Regelvorlagen zur Analyse ausgewählt werden. In Listing 7 sind beispielsweise

zwei verschiedene Ressourcen-Attribute gegeben: Einerseits der konkrete initiierende Akteur (*org:resource*), andererseits die Rolle des Akteurs (*org:group*). Typischerweise ist nur eines dieser beiden Attribute vorhanden. Nur in wenigen Fällen ist gar kein Ressourcen-Attribut verzeichnet.

Qualität von Organisationsmodellen

Neben dem Prozess-Ereignisprotokoll wird zur Ableitung bestimmter organisationsbezogener Regeln auch das zu Grunde liegende Organisationsmodell benötigt. Wie Ereignisprotokolle können auch Organisationsmodelle von unterschiedlicher Qualität bzw. Aussagekraft sein. Je nach verwendetem Organisations-Meta-Modell sind dort beispielsweise lediglich Individuen in Gruppen bzw. Rollen aggregiert (*einfach*) oder in anderen Fällen auch komplexere organisatorische Beziehungen abgebildet (*komplex*). Anhand der Charakteristika gegebener Daten können die zu analysierenden Regelvorlagen der organisatorischen Perspektive schließlich identifiziert werden. Tabelle 10 gibt einen Überblick über die analysierbaren Regelvorlagen in Abhängigkeit von gegebenen Ressourcen-Attributen und der Qualität des Organisationsmodells.

Tabelle 10: Auswahl von Regelvorlagen der organisatorischen Perspektive

▼OM/RA►	kein	org:resource	org:group
kein	⊘	*direct*(A,I), *binding*(A_1,A_2) *separate*(A_1,A_2), *caseHandling*	*role*(A,G), *bindingRole*(A_1,A_2) *separateRole*(A_1,A_2)
einfach	⊘	⊕ *role*(A,G)	
komplex	⊘ ⊘	⊕ *capability*(A,RT,G) ⊕ *orgDistM*(A_1,A_2,RT) ⊕ *orgDistS*(A,RT,G)	⊕ *orgDistS*(A,RT,G)

Die Muster in Zeile 1 sind alleine anhand der Ressouren-Attribute im Ereignisprotokoll ableitbar. Hier wird kein Organisationsmodell benötigt. Ist ein einfaches Organisationsmodell gegeben, kann im Fall des *org:resource*-Attributs **zusätzlich** das *role*-Muster abgeleitet werden. Ist ein *komplexes* Organisationsmodell nach Bussler [30] gegeben, so kann auch nach den Mustern *capability*, *orgDistS* und *orgDistM* gesucht werden.

5.3 IDENTIFIKATION VON AGILEN TEILPROZESSEN

Fokussierung auf agile Teile des Prozesses

Neben der Spezifikation der Menge an zu analysierenden Regelvorlagen kann der Suchraum des Verfahrens auch auf andere Weise eingeschränkt werden. Anstatt zu spezifizieren **nach was** gesucht wird, fokussieren wir im folgenden Abschnitt die Frage **worin** gesucht wird. Wie bereits erwähnt bestehen Prozesse in Organisationen häufig sowohl aus strikten als auch aus agilen Teilprozessen. In der Einleitung dieser Arbeit wird gezeigt, dass regelbasierte Process Mining-Verfahren gut zur Analyse agiler Teilprozesse geeignet sind. Für strikte Teilprozesse eignen sich klassische, prozedurale Process Mining-Verfahren besser [11]. Um aussagekräftige Prozessmodelle mit dem vorliegenden Verfahren

erzeugen zu können, müssen daher die Ereignisse der agilen Teile des Prozesses im betrachteten Ereignisprotokoll identifiziert werden. Nur dieser Teil des Ereignisprotokolls wird anschließend mit dem vorliegenden Verfahren analysiert.

Kontextanalyse

Zur Identifikation von agilen Teilprozessen kann die sog. **Kontextanalyse** (*engl. context analysis)* [82] verwendet werden. Bei diesem Verfahren werden die Ereignisse des betrachteten Ereignisprotokolls in zwei separate Mengen aufgeteilt. Die erste Menge enthält alle Ereignisse die in einem strukturierten, d.h. strikten, Kontext auftreten, wohingegen die zweite Menge diejenigen Ereignisse enthält, die in einem unstrukturierten, d.h. agilen, Kontext auftreten. Der erste Schritt besteht darin für jedes Ereignis die Anzahl an verschiedenen Vorgängern und Nachfolgern im Ereignisprotokoll zu bestimmen. Darauf basierend wird ein Ereignis mit einer großen Anzahl an sowohl Vorgängern als auch Nachfolgern als unstrukturiert klassifiziert, wohingegen ein Ereignis mit einer kleinen Anzahl an Vorgängern oder Nachfolgern als strukturiert klassifiziert wird. Diese Klassifikation basiert auf den folgenden Schlussfolgerungen:

- Besitzt ein Ereignis eine **große Anzahl an Vorgängern** und eine **große Anzahl an Nachfolgern**, dann existieren offensichtlich nur wenige Einschränkungen bezgl. dessen, wann genau das betroffene Ereignis eintreten kann. Das Ereignis tritt daher wahrscheinlich in einem agilen Prozesskontext auf, der sich besser regelbasiert modellieren lässt.
- Besitzt ein Ereignis hingegen lediglich eine **kleine Anzahl an verschiedenen Vorgängern und Nachfolgern**, dann ist es eher wahrscheinlich, dass das Ereignis in einem strikten Prozesskontext auftritt, der sich besser prozedural modellieren lässt, beispielsweise als Sequenzfluss oder als Auswahl zwischen wenigen Alternativen.
- Im Fall, dass ein Ereignis eine **kleine Anzahl an Vorgängern** hat, jedoch **viele verschiedene Nachfolger** besitzt, ist es wahrscheinlich, dass das Ereignis das letzte Element einer strikten Sequenz ist, welcher ein agiler Teilprozess folgt. Es macht daher Sinn, das Ereignis als strikt zu klassifizieren.
- Im Fall, dass ein Ereignis nur **wenige verschiedene Nachfolger** besitzt, jedoch eine **große Anzahl an Vorgängern**, ist es wahrscheinlich, dass das Ereignis das erste Element einer strukturierten Sequenz darstellt, welcher ein agiler Teilprozess voranging. Es macht daher auch hier Sinn, das Ereignis als strikt zu klassifizieren.

Menge an Ereignissen der agilen Teilprozesse

Die Kontextanalyse liefert auf diese Weise zwei Mengen an Ereignissen: Die Menge an strukturierten und unstrukturierten Ereignissen. Anhand dieser beiden Mengen können strikte und agile Teilbereiche im Ereignisprotokoll identifiziert werden. Hierzu wird das Ereignisprotokoll durchlaufen und immer dann ein neuer Bereich begonnen, wenn ein Ereignis nicht der gleichen Menge angehört wie dessen Vorgänger.

5.4 ERZEUGUNG VON RELEVANTEN REGELKANDIDATEN

Viele Entitäten in realen Prozessen

Die Inhalte des folgenden Abschnitts basieren größtenteil auf der eigenen Publikation des Autors [117]. Wie bereits erwähnt, können in realen Prozessen sehr viele verschiedene Entitäten auftreten, d.h. beispielsweise, dass viele verschiedene Aktivitäten durchzuführen sind, wobei eine Vielzahl an verschiedenen Personen bzw. Benutzergruppen beteiligt ist. Bislang verwenden wir die vollständige Menge sämtlicher im Ereignisprotokoll auftretenden Entitäten zur Erzeugung der Regelkandidaten. Dabei werden alle möglichen Parameterkombinationen gebildet. Es wird bislang nicht beachtet, ob eine Parameterkombination überhaupt sinnvoll bzw. interessant ist.

Bislang naive Erzeugung von Regelkandidaten

Zur sinnvollen Reduktion der Anzahl an zu überprüfenden Regelkandidaten beschäftigt sich der folgende Abschnitt mit der Suche nach **interessanten, sinnvollen Parameterkombinationen**, welche durch eine Vorverarbeitung des betrachteten Ereignisprotokolls abgeleitet werden können. Ziel ist es zu analysieren, welche Entitäten, d.h. welche Parameterkombinationen, für die Überprüfung überhaupt relevant sind. Regelkandidaten, die durch Instanziierung mit Parameterkombinationen entstanden sind, die nie zusammen in einer Spur auftreten, sind in jeder Prozessinstanz trivial erfüllt. Deren Überprüfung kostet die gleiche Rechenleistung wie die jeder anderen Regel, erzielt aber keinerlei Erkenntnisgewinn. Zusätzlich zu den häufig auftretenden Parameterkombinationen können interessante Kombinationen **manuell** durch Analysten hinzugefügt werden.

Suche nach sinnvollen Parameterkombinationen

Automatisierte Auswahl von Parameterkombinationen

Wir betrachten zuerst die automatisierte Auswahl häufig auftretender Parameterkombinationen. Zur **Abstraktion** von nicht bzw. nur sehr selten auftretenden Parameterkombinationen verwenden wir eine klassische Data Mining-Methode zur Ableitung **häufiger Muster** in Transaktionsdatenbanken. Das gesamte Verfahren ist in Abbildung 31 dargestellt und besteht aus drei Schritten:

- (1) Transformation des gegebenen Prozessereignisprotokolls zu Transaktionsdatenbanken. Die Art und Anzahl der Transaktionsdatenbanken ist dabei abhängig von der gewählten Menge an Regelvorlagen.
- (2) Ableiten von häufigen Mustern (*engl. frequent patterns)* in diesen Transaktionsdatenbanken. Die abgeleiteten Muster stellen häufig zusammen auftretende Parameterkombinationen dar.
- (3) Erzeugung von relevanten Regelkandidaten durch Instanziierung von Regelvorlagen mit den abgeleiteten, häufigen Parameterkombinationen.

Transaktionsdatenbanken

Eine Transaktionsdatenbank D besteht aus einer Menge von Transaktionen t. Eine **Transaktion** t ist hierbei zusammengesetzt aus Objekten (*engl. Items*), die zusammen auftreten, unabhängig davon ob es sich um materielle (z.B. Artikel im Supermarkt) oder immaterielle (Dienstleistungen, Informationen) Objekte handelt. Ein anschauliches Beispiel für eine Datenbank von Transaktionen ist die Datensammlung, die sich aus den Scanvorgängen an den Kassen eines Supermarktes ergibt. Eine Transaktion besteht in diesem Fall aus allen Artikeln,

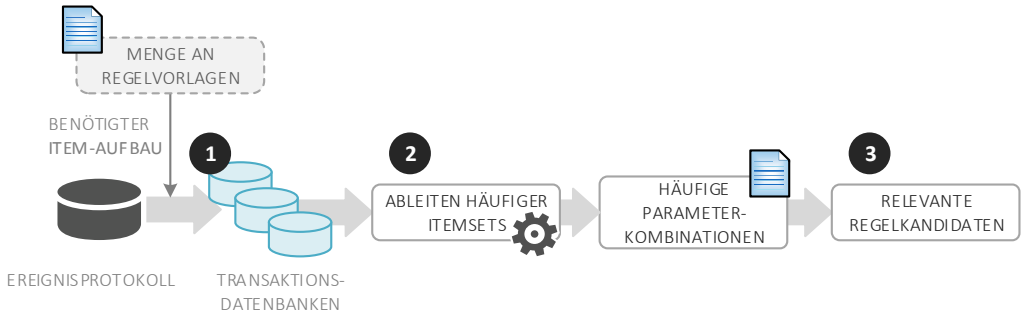


Abb. 31: Verfahren zur Ableitung relevanter Regelkandidaten

die ein Kunde zu einem Zeitpunkt gekauft hat. Die Grundlage für eine tiefergehende Analyse von Transaktionsdatenbanken ist das Bestimmen von Objektmengen, so genannten *Itemsets*, die häufig in dieser Datenbank auftreten (*engl. frequent itemsets*), wobei die Itemsets auch als k-Itemsets bezeichnet werden. k bezeichnet hier die Anzahl der Elemente des Itemsets. Hierfür wird zunächst der Support $supp(X)$ für ein Itemset X berechnet, welcher den Anteil von Transaktionen darstellt, die X enthalten. Von einem **häufigen Itemset** X spricht man, wenn $supp(X)$ größer bzw. gleich einem festgelegten Schwellenwert minSupp ist, also $supp(X) \geqslant minSupp$. Der Fokus ist hinsichtlich großer Datenmengen auf den Aufgabenbereich der Bestimmung häufiger Itemsets (*engl. frequent itemset mining*) gerichtet, da für diese Aufgabe zunächst alle möglichen Itemsets, die sich aus dem Datenbestand ergeben, zu bestimmen sind.

Häufige Itemsets

Um nur relevante Regelkandidaten zu erzeugen, wollen wir nun mit diesem Verfahren häufige Parameterkombinationen, d.h. häufige Itemsets, aus dem betrachteten Ereignisprotokoll ableiten. Hierzu ist eine Transformation des Ereignisprotokolls in potentiell mehrere Transaktionsdatenbanken erforderlich, deren Transaktionen unterschiedliche Struktur besitzen.

5.4.1 Transaktionsdatenbankerzeugung

Zur Klärung der Frage warum, je nach gewählten Regelvorlagen, potentiell mehrere unterschiedliche Transaktionsdatenbanken notwendig sind, betrachten wir die Parametertypen des *sequence(Activity, Activity)*-Makros und des *direct(Activity, Identity)*-Makros. Relevante Parameterkombinationen für das *sequence*-Makro entsprechen offensichtlich häufigen Itemsets der Form {(*Activity*), (*Activity*)}, d.h. 2-Itemsets, wohingehen für die *direct*-Vorlage häufige Itemsets der Form {(*Activity*, *Identity*)}, d.h. 1-Itemsets, benötigt werden. Die Anzahl an benötigten Elementen pro Itemset ist abhängig von der Anzahl an Aktivitäten.

Parametertypen von Regeln

Es ist zu beachten, dass sich vor allem auch die Form der Items unterscheidet. Während beim *sequence*-Makro Items der Form *(Activity)* benötigt werden, muss ein Item beim *direct*-Makro den Aufbau *(Activity, Identity)* aufweisen. Es zeigt sich, dass für bestimmte Regelvorlagen Itemsets mit unterschiedlicher Anzahl an Items sowie unterschiedlicher Form der Items benötigt werden. Dies lässt

Form von Items

sich durch Erzeugung mehrerer verschiedener Transaktionsdatenbanken mit Items entsprechender Form erreichen.

Benötigte Anzahl und Form von Items

Die Anzahl und die Form der benötigten Transaktionsdatenbanken lässt sich aus der Menge der spezifizierten Regelvorlagen ableiten. Für Regelvorlagen, die rein die verhaltensorientierte Perspektive betreffen (z.B. *sequence*, *directSequence*), besteht ein Item lediglich aus einer Aktivität. Für Regelvorlagen, welche die organisatorische Perspektive einbeziehen, werden Items benötigt, die zusätzlich zur Aktivität die durchführende Person bzw. Gruppe enthalten. Allgemein betrachtet, lässt sich aus den freien Parametern einer Regelvorlage direkt die notwendige Form eines Items und die Anzahl der benötigten Elemente in einem Itemset ablesen.

Eine exemplarische Beschreibung der benötigten Itemsets und der Form der Items für die in Abschnitt 4.2 definierten Regelvorlagen erfolgt in den folgenden Abschnitten. Dabei wird ersichtlich, dass der Aufbau der benötigten Items von den Typen der Parameter bestimmt wird.

Itemsets für Regelvorlagen der verhaltensorientieren Perspektive

Itemsets für verhaltensorientierte Regeln

Tabelle 11 zeigt die benötigten Itemsets und deren Form für die Regelvorlagen der **verhaltensorientieren Perspektive**. Eine *sequence*-Regel setzt beispielsweise zwei verschiedene Aktivitäten in Beziehung, daher werden 2-Itemsets der Form {*(Activity), (Activity)*} benötigt. Für *once*-Regeln werden hingegen 1-Itemsets der Form {*Activity*} benötigt.

Tabelle 11: Itemsets für Regelvorlagen der verhaltensorientieren Perspektive

Makro	Form des Items	Benötiges Itemset
sequence(A, A)	(Activity)	L_2: {(Activity), (Activity)}
directSequence(A, A)	(Activity)	L_2: {(Activity), (Activity)}
once(A)	(Activity)	L_1: {(Activity)}

Itemsets für Regelvorlagen der organisatorischen Perspektive

Itemsets für organisationsorientierte Regeln

In Tabelle 12 werden die Itemsets für **organisatorische Zuweisungsregeln** betrachtet, welche genau eine Aktivität einbeziehen.

Tabelle 12: Itemsets für Regelvorlagen der Zuweisungsregeln, welche genau eine Aktivität einbeziehen

Makro	Form des Items	Benötiges Itemset
direct(A, I)	(Activity, Identity)	L_1: {(Activity, Identity)}
role(A, G)	(Activity, Group)	L_1: {(Activity, Group)}
capability(A, RT, G)	(Activity, Group)	L_1: {(Activity), (Group)}
orgDistSingle(A, RT, G)	(Activity, Group)	L_1: {(Activity), (Group)}

Da in allen Mustern lediglich eine Aktivität beteiligt ist, werden stets nur 1-Itemsets benötigt. Die *direct*-Vorlage beispielsweise bezieht sich nur auf eine

Aktivität, schließt jedoch Identitäten ein, daher werden 1-Itemsets der Form {*(Activity, Identity)*} benötigt. Die *role*-Vorlage bezieht sich ebenfalls auf eine Aktivität und besitzt einen zweiten freien Parameter vom Typ *Group*. Es werden daher 1-Itemsets der Form {*(Activity, Group)*} benötigt. Die Makros *capability* und *orgDistSingle* besitzen zusätzlich einen freien Parameter vom Typ *RelationType*. Die Anzahl an verschiedenen Beziehungstypen in Organisationsmodellen ist jedoch im Vergleich zur Anzahl an Aktivitäten oder Identitäten/Gruppen sehr klein. Dieser Parameter wird daher nicht berücksichtigt.

In Tabelle 13 werden die Itemsets für organisatorische Zuweisungsregeln betrachtet, welche zwei Aktivitäten einbeziehen. Da in allen Mustern zwei Aktivitäten beteiligt sind, werden stets nur 2-Itemsets benötigt. Die *binding*-Vorlage beispielsweise bezieht sich auf zwei Aktivitäten und enthält keine zusätzlichen freien Variablen. Es werden daher 2-Itemsets der Form {*(Activity), (Activity)*} benötigt.

Tabelle 13: Itemsets für Regelvorlagen der Zuweisungsregeln, welche mehrere Aktivitäten einbeziehen

Makro	Form des Items	Benötiges Itemset
binding(A, A)	(Activity)	L_2: {(Activity), (Activity)}
separate(A, A)	(Activity)	L_2: {(Activity), (Activity)}
orgDistMulti(A, A, RT)	(Activity, Activity)	L_2: {(Activity, Activity)

Itemsets für perspektivenübergreifende Regelvorlagen

Itemsets für perspektivenübergreifende Regeln

In Tabelle 14 sind die Itemsets perspektivenübergreifender Zusammenhänge dargestellt. Da in allen Mustern zwei Aktivitäten beteiligt sind, werden stets 2-Itemsets benötigt. Die *roleSequence*-Vorlage bezieht sich auf zwei Aktivitäten und enthält zusätzlich eine freie Variable vom Type *Group*. Es werden daher 2-Itemsets der Form {*(Activity, Group), (Activity, Group)*} benötigt.

Tabelle 14: Itemsets für Regelvorlagen perspektivenübergreifender Zusammenhänge

Makro	Form des Items	Benötiges Itemset
resourceSequence(A, A, I)	(Activity, Identity)	L_2: {(Activity, Identity), (Activity, Identity)}
roleSequence(A, A, G)	(Activity, Group)	L_2: {(Activity, Group), (Activity, Group)}

Item-Formen für ausgewählte Regelvorlagen

Für die in Abschnitt 4.2 definierte Menge an Regelvorlagen werden daher 3 verschiedene Transaktionsdatenbanken mit den entsprechenden Item-Formen *(Activity)*, *(Activity, Identity)* und *(Activity, Group)* benötigt. Die konkrete Transaktionsdatenbankerzeugung für ein gegebenes Ereignisprotokoll wird nun an einem Beispiel erläutert.

BEISPIEL (TRANSAKTIONSERZEUGUNG AUS EREIGNISPROTOKOLL) Gegeben sei die exemplarische Menge Θ an zu analysierenden Regelvorlagen:

$$\Theta = \{\textit{sequence}(A_1,A_2), \textit{direct}(A,I), \textit{resourceSequence}(A_1,A_2,I)\}$$

Aus der Analyse der freien Parameter dieser Regelvorlagen ergibt sich, dass 2 verschiedene Transaktionsdatenbanken mit den Item-Formen *(Activity)* und *(Activity, Identity)* erforderlich sind. Des Weiteren sei ein Ereignisprotokoll Φ gegeben mit $|\Phi| = 5$ und den Spuren

σ_1: $\{\mathbf{s}(a_1, i_1), \mathbf{s}(a_2, i_2), \mathbf{c}(a_2, i_2), \mathbf{c}(a_1, i_1), \mathbf{s}(a_3, i_1), \mathbf{c}(a_3, i_1)\}$
σ_2: $\{\mathbf{s}(a_1, i_1), \mathbf{c}(a_1, i_1), \mathbf{s}(a_4, i_3), \mathbf{c}(a_4, i_3)\}$
σ_3: $\{\mathbf{s}(a_1, i_1), \mathbf{c}(a_1, i_1), \mathbf{s}(a_3, i_1), \mathbf{c}(a_3, i_1), \mathbf{s}(a_2, i_2), \mathbf{c}(a_2, i_2)\}$
σ_4: $\{\mathbf{s}(a_2, i_2), \mathbf{c}(a_2, i_2), \mathbf{s}(a_1, i_1), \mathbf{c}(a_1, i_1), \mathbf{s}(a_3, i_1), \mathbf{c}(a_3, i_1)\}$
σ_5: $\{\mathbf{s}(a_2, i_2), \mathbf{c}(a_2, i_2), \mathbf{s}(a_4, i_4), \mathbf{s}(a_1, i_1), \mathbf{c}(a_4, i_4), \mathbf{c}(a_1, i_1)\}$

Betrachten wir zunächst die Erzeugung der Transaktionsdatenbank D_1, die nur Items der Form *(Activity)* enthalten soll. Für jede Spur $\sigma \in \Phi$ wird eine Transaktion $t \in D_1$ erzeugt, D_1 enthält daher 5 Transaktionen. Für jede auftretende Aktivität a in der aktuell betrachteten Spur σ_x wird dabei genau ein Item (a) erzeugt. Eine Transaktion t_x beinhaltet somit die duplikatsfreie Menge an Aktivitäten, die in σ_x auftreten. Diese Transformation führt zu der Transaktionsdatenbank D_1:

D_1: $\{a_1,a_2,a_3\}, \{a_1,a_4\}, \{a_1,a_3,a_2\}, \{a_2,a_1,a_3\}, \{a_2,a_4,a_1\}$

Auf ähnliche Weise wird auch die zweite Transaktionsdatenbank D_2 erzeugt. Items von D_2 haben die Form *(Activity, Identity)*. Auch hier wird für jede Spur $\sigma \in \Phi$ eine Transaktion $t \in D_2$ erzeugt. Für jede auftretende Kombination aus Aktivität a und entsprechender Identität i wird genau ein Item (a, i) erzeugt. Eine Transaktion t_x beinhaltet daher die duplikatsfreie Menge an Aktivitäten und entsprechenden Identitäten, die in den Ereignissen von σ_x auftreten. Diese Transformation führt zu D_2:

D_2: $\{(a_1,i_1),(a_2,i_2),(a_3,i_1)\}, \{(a_1,i_1),(a_4,i_3)\}, \{(a_1,i_1),(a_3,i_1),(a_2,i_2)\}$,
$\{(a_2,i_2),(a_1,i_1),(a_3,i_1)\}, \{(a_2,i_2),(a_4,i_4),(a_1,i_1)\}$

In den auf diese Weise erzeugten Transaktionsdatenbanken wollen wir nun häufige Itemsets bestimmen. Diese stellen die interessanten Parameterkombinationen dar, anhand derer anschließend die relevante Menge an Regelkandidaten generiert wird.

5.4.2 Ableiten häufiger Parameterkombinationen

Sei n die Menge aller Items, die in einer Transaktionsdatenbank mindestens einmal auftreten. Dann ist die Menge aller möglichen Itemsets die Potenzmenge von I ohne die leere Menge und somit $|I| = 2^n - 1$. Der **Apriori-Algorithmus** [14] ist ein Verfahren der Assoziationsanalyse, um häufige Itemsets aus Transaktionsdatenbanken zu extrahieren, d.h. die Menge aller häufigen Itemsets L, wobei für alle $l \in L$ gilt $supp(l) \geqslant minSupp$. Die Grundlage dieses Algorithmus ist das **Apriori-Prinzip**, welches besagt, dass auch die Teilmengen häufiger Itemsets wieder häufig sind. Zusätzlich gilt, dass der Support eines Itemsets nicht größer sein kann als der größte Support der Teilmengen.

Apriori-Algorithmus

Apriori-Prinzip

Algorithm 1 Apriori-Algorithmus

```
1:  input: T - Transaktionen
2:  input: minSupp - Support-Schwellenwert für häufige Itemsets
3:  output: Menge an häufigen Itemsets
4:  procedure Apriori(T, minSupp)
5:  L_1={frequent 1-Itemsets};
6:  for (k = 2; L_{k-1} ≠ ∅; k++) do
7:    C_k=APRIORI − GEN(L_{k-1});
8:    for all transactions t ∈ T do
9:      C_t = SUBSET(C_k, t);
10:     for all candidates c ∈ C_t do
11:       c.count++;
12:     end for
13:   end for
14:   L_k = {c ∈ C_k | c.count ⩾ minSupp}
15: end for
16: return ⋃ L_k
```

Der Apriori-Algorithmus (Algorithmus 1) zählt in der ersten Iteration zunächst alle häufigen 1-Itemsets durch einen Scan der Transaktionsdatenbank. Die nachfolgenden Iterationen bestehen jeweils aus zwei Phasen. In der ersten Phase wird aus den im vorherigen Durchlauf berechneten häufigen $(k-1)$-Itemsets mit dem Apriori-Gen-Algorithmus die Kandidatenmenge der häufigen k-Itemsets berechnet (Zeile 7). In der zweiten Phase wird der Support für die Kandidaten in C_k gezählt. Itemsets der Länge k mit einem Support größer gleich dem Schwellenwert $minSupp$ werden zum Abschluss der Menge der häufigen k-Itemsets hinzugefügt (Zeile 14). Am Ende gibt der Algorithmus schließlich die Vereinigung aller häufigen Itemsets zurück (Zeile 16).

Apriori-Gen-Algorithmus

Der Algorithmus zur Erzeugung neuer, größerer Itemsets (Algorithmus 2) benutzt die Menge L_{k-1}, d.h. die Menge aller häufigen $(k-1)$-Itemsets, und konstruiert daraus eine Obermenge von Itemsets der Länge k. In den Zeilen 4 bis 7 werden diese erzeugt, indem zunächst zwei häufige $(k-1)$-Itemsets p und q gewählt werden, die auf den Positionen $1, ..., k-2$ identisch sind und

Algorithm 2 APRIORI-GEN-Algorithmus

1: input: L_{k-1} - häufige Itemsets der Länge $k-1$
2: output: C_k - Kandidaten für häufige Itemsets der Länge k
3: **procedure** APRIORI-GEN(L_{k-1})
4: insert into C_k
5: select $p.item_1, p.item_2, ..., p.item_{k-1}, q.item_{k-1}$
6: from $L_{k-1}p, L_{k-1}q$
7: where $p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$;
8: **for all** itemsets $c \in C_k$ **do**
9: **for all** $(k-1)$-subsets s of c **do**
10: **if** $s \notin L_{k-1}$ **then**
11: delete c from C_k
12: **end if**
13: **end for**
14: **end for**
15: **return** C_k

sich in der $(k-1)$-ten Position unterscheiden, das $(k-1)$-te Element von p jedoch kleiner als das von q ist. Die Positionen $1, ..., (k-2)$ werden nun mit den zwischen p und q gleichen Elementen besetzt, die $(k-1)$-te Position mit dem $(k-1)$-ten Element aus p und das k-te Element mit dem aus q. Im Anschluss an die Erzeugung der Itemsets ist abschließend die Apriori-Eigenschaft zu prüfen, also ob die Kandidaten häufige $(k-1)$-Itemsets enthalten. Dadurch lässt sich bereits die Menge der Itemsets reduzieren, deren Support zu berechnen ist. Hierfür werden alle Kandidaten durchlaufen (Zeile 8), von denen wiederum alle $(k-1)$-Teilmengen durchlaufen werden. Wenn nun eine dieser Teilmengen nicht häufig ist, d.h. nicht in L_{k-1} enthalten ist, so folgt aus der Apriori-Eigenschaft, dass das k-Itemset nicht häufig sein kann.

Support-Wert eines Itemsets

Bevor wir obiges Verfahren anwenden, ist es notwendig den Support-Wert eines Itemsets X für Transaktionsdatenbanken zu definieren, die aus der Transformation eines Ereignisprotokolls entstanden sind. Sei D die aus einem Ereignisprotokoll Φ entstandene Transaktionsdatenbank. Sei T_X die Menge derjenigen Transaktionen t, die sämtliche Items $x \in X$ des Itemsets X enthalten. Dann ist der Support eines Itemsets X in D definiert durch:

$$supp(X) = \frac{|T_X|}{|D|}, \text{mit } T_X = \{t \in D | \forall_{x \in X}\, x \in t\} \tag{4}$$

Da für jede Spur des Ereignisprotokolls, d.h. jede Prozessinstanz, eine Transaktion generiert wird, spiegelt der Support-Wert eines Itemsets daher das Verhältnis an Prozessinstanzen in denen das Itemset auftritt, zur Gesamtanzahl an Instanzen wider. Die Anwendung des Algorithmus wird nun an einem Beispiel erläutert.

BEISPIEL (APRIORI-ALGORITHMUS ZUM ABLEITEN HÄUFIGER ITEMSETS) Wir betrachten wieder die bereits generierte Transaktionsdatenbank D_1:

D_1: $\{a_1,a_2,a_3\}$, $\{a_1,a_4\}$, $\{a_1,a_3,a_2\}$, $\{a_2,a_1,a_3\}$, $\{a_2,a_4,a_1\}$

Ziel ist es häufige Itemsets in D_1 zu finden, die einen Support-Wert von mind. 50% aufweisen, d.h. die in mind. 50% aller Transaktionen und daher aller Prozessinstanzen auftreten. Tabelle 15 gibt einen Überblick über die Anwendung des Algorithmus auf D_1. Der Algorithmus beginnt damit den Support für alle 1-Itemsets zu berechnen, d.h. den Support aller auftretenden einzelnen Aktivitäten. Unter der Kandidatenmenge C_1 ist das 1-Itemset mit der Aktivität a_4, dessen Support lediglich 40% beträgt, was unter $minSupp$ liegt. Häufige 1-Itemsets L_1 sind daher nur $\{a_1, a_2, a_3\}$. In der nächsten Iteration werden Kandidatenmengen C_2 der Größe 2 betrachtet. Da der Support von a_4 bereits kleiner als der definierte Schwellenwert ist, sind auch alle größeren Itemsets die a_4 enthalten nicht häufig. Alle Kandidaten in C_2 werden daher nur aus häufigen 1-Itemsets erzeugt. Die Support-Werte der 3 erzeugten 2-Itemsets erfüllen $minSupp$ und sind daher alle in L_2. Eine Iteration weiter erhalten wir lediglich noch ein Itemset in C_3 der Größe 3, welches ein häufiges Itemset darstellt und daher in L_3 ist. Der Algorithmus terminiert an dieser Stelle und liefert die Menge an häufigen Itemsets $L = L_1 \cup L_2 \cup L_3$.

5.4.3 Erzeugung von Regelkandidaten aus häufigen Itemsets

Erzeugung von relevanten Regelkandidaten

Aus der resultierenden Menge an häufig auftretenden Itemsets können nun relevante Parameterkombinationen zur Regelkandidatenerzeugung abgeleitet werden. Hierzu betrachten wir die benötigten Itemset-Größen für ausgewählte Regelvorlagen. Zur Generierung von Regelkandidaten der *sequence*-Vorlage werden beispielsweise die häufigen 2-Itemsets des Typs *(Activity)* benötigt. Diese liegen in der Menge L_2 vor. Die *sequence*-Regelvorlage wird hier mit allen möglichen Permutationen der häufigen 2-Itemsets instanziiert. Aus dem häufigen Itemset $\{a_1, a_2\}$ werden auf diese Weise beispielsweise die Regelkandidaten *sequence*(a_1,a_2) und *sequence*(a_2,a_1) erzeugt. Durch die Verwendung der häufigen Parameterkombinationen kann die Anzahl an Regelkandidaten deutlich reduziert werden. Im Beispiel werden durch alle möglichen Permutationen der häufigen 2-Itemsets insgesamt 6 verschiedene *sequence*-Regelkandidaten er-

Reduktion der Anzahl an Regelkandidaten

Tabelle 15: Anwendung des Apriori-Algorithmus auf D_1

C_1	**supp(X)**	$\in L_1$	C_2	**supp(X)**	$\in L_2$	C_3	**supp(X)**	$\in L_3$
a_1	100	✓	$\{a_1, a_2\}$	80	✓	$\{a_1, a_2, a_3\}$	60	✓
a_2	80	✓	$\{a_1, a_3\}$	60	✓			
a_3	60	✓	$\{a_2, a_3\}$	60	✓			
a_4	40							

zeugt. Im Gegensatz dazu würden bei einem naiven Ansatz, bei dem die Regelvorlage mit allen möglichen Kombinationen instanziiert wird, $4^2 = 16$ Regelkandidaten erzeugt.

5.4.4 Manuelles Hinzufügen von Parameterkombinationen

Im vorherigen Abschnitt haben wir das gegebene Ereignisprotokoll zu Transaktionsdatenbanken transformiert und daraus *automatisiert* häufig zusammen auftretende Entitäten abgeleitet. Auf Basis dieser häufigen Itemsets werden nur Regelkandidaten generiert, die auch wirklich relevant sind, d.h. deren Parameter auch tatsächlich und häufig genug zusammen in den aufgezeichneten Prozessinstanzen auftreten.

Interessante, aber nicht häufige Kombinationen

Es kann jedoch zusätzlich Parameterkombinationen geben, die interessant aber nicht häufig sind, d.h. nur sehr selten zusammen in den aufgezeichneten Prozessinstanzen auftreten. Interessante, jedoch nicht häufige Parameterkombinationen müssen daher manuell von Analysten ausgewählt und hinzugefügt werden können. Auch für diese Parameter werden anschließend Regelkandidaten erzeugt und während des Analysevorgangs überprüft. Abbildung 32 visualisiert die in den vorherigen Abschnitten eingeführten Konzepte anhand von *sequence*-Regeln. Anstatt also alle möglichen Regelkandidaten zu generieren und zu überprüfen wird nur eine relevante Teilmenge betrachtet. Diese setzt sich zusammen aus den automatisiert abgeleiteten, häufig zusammen auftretenden Parameterkombinationen (1) sowie aus den benutzerdefinierten, als interessant erachteten Kombinationen (2).

Manuelles Hinzufügen

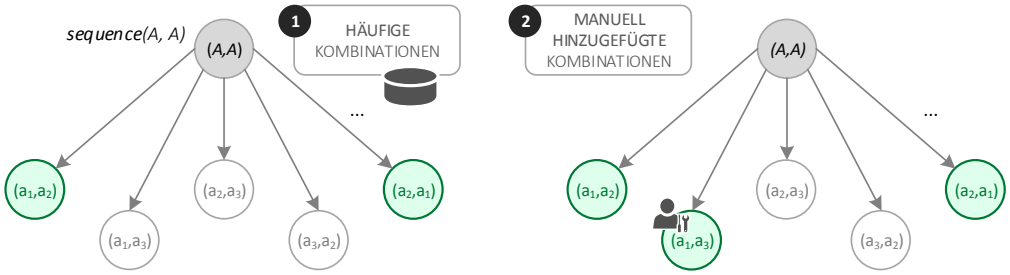


Abb. 32: Auswahl relevanter Parameterkombinationen

5.5 ZUSAMMENFASSUNG

In diesem Kapitel wurde betrachtet, wie das vorgestellte Verfahren zur automatisierten Erzeugung von regelbasierten Prozessmodell konfiguriert werden kann und der Suchraum durch eine Vorverarbeitung des betrachteten Ereignisprotokolls sinnvoll eingeschränkt werden kann.

Die **Konfiguration** des Verfahrens erfolgt dabei durch die Auswahl einer bestimmten Menge an zu analysierenden Regelvorlagen. Durch diese Menge wird

sowohl die Komplexität der Analyse festgelegt, als auch die Aussagekraft des resultierenden Modells. Das durch die Analyse einer bestimmten Menge an Regelvorlagen generierte Prozessmodell stellt nur eine spezielle Sicht auf das Ereignisprotokoll dar. Welche Regelvorlagen analysiert werden sollen, sollte daher eine bewusste Wahl des Analysten sein.

Durch eine **Vorverarbeitung** des Ereignisprotokolls kann eine sinnvolle Einschränkung des Suchraums erreicht werden. Prozesse bestehen in der Realität häufig sowohl aus strikten als auch aus agilen Teilprozessen. Da das vorliegende Verfahren vorwiegend für die Analyse von Ereignisprotokollen agiler Prozesse geeignet ist, ist eine Einschränkung der Analyse auf agile Teilbereiche des Ereignisprotokolls sinnvoll. Durch das Verfahren der **Kontextanalyse** können die Ereignisse der agilen Teile des betrachteten Prozesses identifiziert und isoliert betrachtet werden.

Des Weiteren kann durch eine Vorverarbeitung des Ereignisprotokolls auch die Anzahl an zu überprüfenden Regelkandidaten eingeschränkt werden. Bislang wurde nicht beachtet, ob eine Parameterkombination eines Regelkandidaten überhaupt sinnvoll bzw. interessant ist. Zur sinnvollen Reduktion der Anzahl an zu überprüfenden Regelkandidaten wurde daher ein Verfahren erläutert, mit dem **häufige Parameterkombinationen**, d.h. häufige Itemsets, aus dem betrachteten Ereignisprotokoll ableitet werden können. Auf Basis dieser Menge können dann relevante Regelkandidaten erzeugt werden. Durch die Verwendung der häufig auftretenden Parameterkombinationen kann die Anzahl an Regelkandidaten reduziert werden, was die Komplexität der Analyse deutlich verringert.

6 NACHBEARBEITUNG ERZEUGTER MODELLE

INHALT

Die in den vorherigen Kapiteln vorgestellen Methoden ermöglichen die automatisierte Erzeugung von regelbasierten DPIL-Prozessmodellen. Die praktische Anwendbarkeit des Verfahrens wird dabei durch eine effiziente Implementierung sowie durch eine Vorverarbeitungs- und Konfigurationsphase unterstützt. Da extrahierte Modelle zu Dokumentationszwecken herangezogen werden und häufig Grundlage für Diskussionen sind, wollen wir uns im folgenden Kapitel der Verständlichkeit und Lesbarkeit der abgeleiteten Modelle widmen.

6.1 MOTIVATION

Viele Regeln erschweren Verständlichkeit

Nachbearbeiten von Modellen

Je mehr Modellierungselemente, d.h. Regeln, ein Modell enthält, desto schwerer fällt es menschlichen Analysten, den Sinn des Modells zu erfassen. Zur Steigerung der Verständlichkeit extrahierter Modelle sollte die Anzahl an notwendigen Regeln zur Beschreibung des Prozesses möglichst minimiert werden. Die Motivation dieses Kapitels ist daher die Nachbearbeitung abgeleiteter Modelle durch das Entfernen bzw. "Stutzen" (*engl. Pruning*) unnötiger, redundanter Regeln. Betrachten wir hierzu folgende Prozessmodelle:

Listing 8: Beispiel-Modell

```
process Example {
   task a1
   task a2
   task a3

   ensure directSequence(a1,a2)
   ensure sequence(a1,a2)
   ensure sequence(a2,a3)
   ensure sequence(a1,a3)
}
```

Listing 9: Nachbearbeitetes Modell

```
process Example {
   task a1
   task a2
   task a3

   ensure directSequence(a1,a2)

   ensure sequence(a2,a3)

}
```

Das Prozessmodell in Listing 8 wird durch die Analyse der *directSequence*- und der *sequence*-Regelvorlagen abgeleitet. Dem in Kapitel 4 erläuterten Verfahren folgend, enthält das abgeleitete Modell all diejenigen Regeln, die abhängig von den Schwellenwerten $minConf_H$ und $minConf_S$ in den Spuren des betrachteten Ereignisprotokoll eingehalten werden. Dabei wird jedoch nicht betrachtet, ob eine abgeleitete Regel bereits in einer anderen Regel enthalten ist oder ob eine abgeleitete Regel aus anderen Regeln gefolgert werden kann. Die vier Regeln des Prozessmodells besagen beispielsweise, dass Aktivität a_1 direkt vor Aktivität a_2 und a_2 irgendwann vor Aktivität a_3 durchgeführt werden muss. Zwei der vier Regeln sind dabei redundant, welche im Modell in Listing 9 entfernt wurden. Die Aussagekraft des Modells bleibt jedoch gleich. Die *sequence*(a_1,a_2)-Regel (*"irgendwann zuvor"*) ist in der *directSequence*(a_1,a_2)-Regel (*"direkt zuvor"*) bereits enthalten, liefert keinerlei zusätzliche Information und kann daher entfernt werden. Manche Regeln sind also **stärker** bzw. **spezieller** als andere Regeln. Werden zwei Regeln mit gleichen Entitäten extrahiert, kann in vielen Fällen die schwächere Regel entfernt werden, ohne die Aussagekraft des Modells zu verändern. Des Weiteren ist die *sequence*(a_1,a_3)-Regel redundant. Sie lässt sich **transitiv** aus den abgeleiteten Regeln *sequence*(a_1,a_2) und *sequence*(a_2,a_3) folgern, d.h. wenn a_1 vor a_2 und a_2 vor a_3 durchgeführt werden muss, dann muss auch a_1 vor a_3 durchgeführt werden. Bestimmte Mengen abgeleiteter Regeln bilden gerichtete Graphen. Durch die Analyse dieser Graphen können redundante, transitiv ableitbare Regeln identifiziert und anschließend entfernt werden. Dieser Vorgang wird als *transitive Reduktion* bezeichnet. In den folgenden Abschnitten betrachten wir die Nachbearbeitung abgeleiteter Modelle durch Anwendung der beschriebenen Methoden. Auf diese Weise wird die Verständlichkeit der betrachteten Modelle durch das Entfernen redundanter Regeln verbessert.

Schwächere in stärkeren Regeln enthalten

Transitiv ableitbare Regeln

6.2 ENTFERNEN VON REGELN AUF BASIS VON REGELHIERARCHIEN

Wir betrachten zuerst die Situation in der verschiedene Regelvorlagen analysiert werden und abgeleitete Modelle stärkere und schwächere Regeln beinhal-

ten, welche die gleichen Entitäten betreffen. Ziel ist es, bestimmte Regeln aus dem betrachteten Modell zu entfernen, wenn diese bereits in anderen, stärkeren Regeln enthalten sind. Um bestimmen zu können, welche Regel stärker bzw. schwächer ist, müssen die zu analysierenden Regelvorlagen in **Regelhierarchien** angeordnet werden. Dies erfordert, ebenso wie die Definition von Regelvorlagen, Expertise und muss manuell von Modellierungsexperten vorgenommen werden. In Fällen, in denen benutzerdefinierte Regeltypen von Analysten definiert werden, müssen diese in existierende Hierarchien manuell eingeordnet werden. Zur Definition von Regelhierarchien wird für die folgenden Abschnitte die *Dominator*-Relation $\rightarrow_{dom}$ zwischen zwei Regeln r_1 und r_2 eingeführt. Ist eine Regel r_1 stärker als eine Regel r_2, so gilt $r_1 \rightarrow_{dom} r_2$.

Regelhierarchien

Dominator-Relation

6.2.1 Verhaltensorientierte Perspektive

Die Regelhierarchie der in dieser Arbeit betrachteten Regelvorlagen der **verhaltensorientierten Perspektive** wurde bereits erläutert. Enthält das extrahierte Prozessmodell sowohl eine *directSequence*-Regel als auch eine *sequence*-Regel, welche die gleichen Aktivitäten a_1 und a_2 betreffen, so ist die *sequence*-Regel redundant und kann entfernt werden. Es gilt daher ***directSequence***$(a_1,a_2) \rightarrow_{dom}$ ***sequence***(a_1,a_2). Eine Regelhierachie für weitere Regeltypen der verhaltensorientierten Perspektive definieren Maggi et al. in [78].

Regelhierarchie verhaltensorientierter Regeln

6.2.2 Organisatorische Perspektive

Regelhierarchien existieren jedoch auch für die Regeln der **organisatorischen Perspektive**. In dieser Arbeit werden Regelhierarchien für die wichtigsten, etablierten organisatorischen Prozessmuster (*Workflow Resource Patterns)* definiert. Hier müssen wir zwischen den Zuweisungsregeln unterscheiden, welche eine und welche zwei Aktivitäten betreffen. Das Muster der Fallbehandlung, repräsentiert durch die *caseHandling*-Vorlage, fordert, dass alle Aktivitäten einer Instanz von der gleichen Identität durchgeführt werden und nimmt daher eine spezielle Position ein, da es alle Aktivitäten betrifft. Wird das Muster der Fallbehandlung entdeckt, sind alle anderen organisatorischen Zuweisungsregeln redundant. Die *caseHandling*-Regel ist daher für alle organisatorischen Zuweisungsregeln die stärkste Regel.

Regelhierarchien organisatorischer Regeln

Zuweisungsregeln mit Bezug auf eine Aktivität

Betrachten wir zuerst die Regelvorlagen, die nur eine Aktivität betreffen. Diese wurden in Abschnitt 4.2.2 definiert und sind in der Menge Θ_2 zusammengefasst.

Zuweisungsregeln einer Aktivität

$\Theta_2 = \{direct(A,I), role(A,G), capability(A,RT,G), orgDistS(A,RT,G)\}$

Dominator-Relation für organisatorische Regeln

Ziel ist es nun, eine Regelhierarchie der organisatorischen Zuweisungsregeln in Θ_2 aufzustellen. Hierzu betrachten wir eine beliebige Aktivität a_1. Die Menge an Identitäten $I(a_1)$, welche Aktivität a_1 im betrachteten Ereignisprotokoll Φ durchgeführt hat, wird beschrieben durch n abgeleitete organisatorische Zuweisungsregeln r_i. Dabei gilt $I(a_1) = I(r_1) \cap I(r_2) \cap ... \cap I(r_n)$, wobei $I(r_i)$ eine Menge an Identitäten darstellt, die spezielle Eigenschaften erfüllen, die durch die Regel r_i definiert werden. Wir vergleichen nun für alle Kombinationen der Regeltypen aus Θ_2 jeweils zwei Regeln r_1 und r_2 und die Mengen an Identitäten $I(r_1)$ bzw. $I(r_2)$, welche diese Regeln erfüllen. Im ersten Abschnitt zeigen wir, dass *direct*-Regeln stärker als alle anderen Regeln in Θ_2 sind.

- ***direct***$(a_1,i_1) \rightarrow_{dom}$ ***role***(a_1,g_1)***:*** Die *direct*-Regel ist spezieller als die *role*-Regel. Ist in einem abgeleiteten DPIL-Modell für eine bestimmte Aktivität a_1 sowohl eine *direct* als auch eine *role*-Regel enthalten, kann die *role*-Regel entfernt werden. Wir zeigen dies in zwei Schritten:

  (i) Sei I_1=I(*direct*) und I_2=I(*role*). Zuerst wird gezeigt, dass $I_1 \subset I_2$. Dies belegen wir durch einen Widerspruchsbeweis. Sei $I_1 = \{i_1\}$ und $I_2 = \{i_2, i_3,...,i_n\}$ mit $i_1 \notin I_2$, da i_1 nicht die Rolle g_1 besitzt. Dann gilt $I(a_1)=I_1 \cap I_2=\emptyset$. Dies stellt einen Widerspruch dar, da die Menge an Identitäten, die a_1 in Φ durchführten, nicht leer sein kann. Daher gilt $I_1 \subset I_2$.

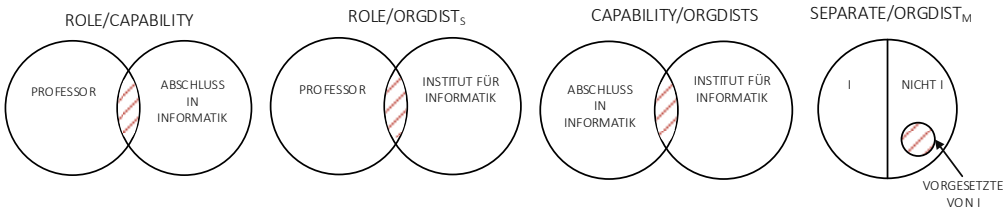
  (ii) Es gilt $I_1 = \{i_1\}$ mit $|I_1| = 1$ und wegen (i) $I_2 = \{i_1,i_2,i_3,...,i_n\}$, insbesondere $i_1 \in I_2$. Die Menge an Identitäten, welche a_1 in Φ durchführten, wird daher beschrieben durch $I(a_1)=I_1 \cap I_2=\{i_1\}=I_1$. Die *direct*-Regel genügt daher zur Beschreibung der Identitäten, die a_1 durchführten.

- ***direct***$(a_1,i_1) \rightarrow_{dom}$ ***capability***(a_1,rt_1,g_1)***:*** Die *direct*-Regel ist spezieller als die *capability*-Regel. Ist in einem abgeleiteten DPIL-Modell für eine bestimmte Aktivität a_1 sowohl eine *direct* als auch eine *capability*-Regel enthalten, kann die *capability*-Regel entfernt werden. Die Beweisführung ist dabei analog.

- ***direct***$(a_1,i_1) \rightarrow_{dom}$ ***orgDistS***(a_1,rt_1,g_1)***:*** Die *direct*-Regel ist spezieller als die *orgDistS*-Regel. Ist in einem abgeleiteten DPIL-Modell für eine bestimmte Aktivität a_1 sowohl eine *direct* als auch eine *orgDistS*-Regel enthalten, kann die *orgDistS*-Regel entfernt werden. Die Beweisführung ist dabei analog.

Vergleichen wir nun die Regeltypen *role*/*capability*, *role*/*orgDistS* und *capability*/*orgDistS*. Für diese Regeltypen kann **keine allgemeingültige Dominator-Relation** definiert werden. Sie befinden sich in der resultierenden Regelhierarchie daher auf der gleichen Ebene. Dies wird durch die Angabe eines Gegenbeispiels für jede Kombination gezeigt.

- ***role***$(a_1,i_1) \not\rightarrow_{dom}$ ***capability***(a_1,rt_1,g_1): Betrachten wir die Situation, in der die Zuweisungsregeln *role*(a_1, *Professor)* und *capability*(a_1, *hasDegree, ComputerScience)* für die Aktivität a_1 abgeleitet wurden. Sei I_1=I(*role*) und



(a) role/capability **(b)** role/orgDistS **(c)** capability/orgDistS **(d)** sep./orgDistM

Abb. 33: Beispiele für die Menge an Identitäten $I(a_1)$

I_2=I(*capability*). Die Menge an Identitäten, die a_1 durchführten, ergibt sich zu $I(a_1)$=$I_1 \cap I_2$. Aktivität a_1 wurde demnach von Identitäten durchgeführt, die eine Rolle *Professor* und *gleichzeitig* die Fähigkeit *Abschluss in Informatik* besitzen. Im Allgemeinen gibt es jedoch auch Professoren, die keinen Abschluss in Informatik haben. Es gilt daher sowohl $I(a_1) = I_1 \cap I_2 \neq I_1$ als auch $I(a_1) = I_1 \cap I_2 \neq I_2$. Zur Beschreibung von $I(a_1)$ sind daher beide Regeln notwendig. Abbildung 33a verdeutlicht den Zusammenhang. Hier ist die Menge $I(a_1)$ als Schnittmenge aus I_1 und I_2 hervorgehoben.

- ***role***$(a_1,i_1) \not\rightarrow_{dom}$ ***orgDistS***(a_1,rt_1,g_1): Hier betrachten wir ein Beispiel, in dem die Zuweisungsregeln *role*(a_1, *Professor)* und *orgDistS*(a_1, *memberOf, DepOfCS)* für die Aktivität a_1 abgeleitet wurden. Sei I_1=I(*role*) und I_2=I(*orgDistS*). Aktivität a_1 wurde demnach von Identitäten durchgeführt, die eine Rolle *Professor* besitzen und *gleichzeitig* Mitglied in der organisatorischen Gruppe *Institut für Informatik* sind. Im Allgemeinen sind nicht alle Professoren auch Mitglied dieser Gruppe. Es gilt daher sowohl $I(a_1) = I_1 \cap I_2 \neq I_1$ als auch $I(a_1) = I_1 \cap I_2 \neq I_2$. Zur Beschreibung von $I(a_1)$ sind daher wiederum beide Regeln erforderlich. In Abbildung 33b ist die Menge $I(a_1)$ als Schnittmenge aus I_1 und I_2 hervorgehoben.

- ***capability***$(a_1,rt_1,g_1) \not\rightarrow_{dom}$ ***orgDistS***(a_1,rt_1,g_1): Als Beispiel betrachten wir zwei abgeleitete Zuweisungsregeln *capability*(a_1, *hasDegree, ComputerScience)* und *orgDistS*(a_1, *memberOf, DepOfCS)* für die Aktivität a_1. Sei I_1=I(*capability*) und I_2=I(*orgDistS*). Aktivität a_1 wurde demnach von Identitäten durchgeführt, die einen *Abschluss in Informatik* besitzen und *gleichzeitig* Mitglied in der organisatorischen Gruppe *Institut für Informatik* sind. Im Allgemeinen hat nicht jeder Angestellte des Instituts für Informatik einen Abschluss in Informatik. Es gilt sowohl $I(a_1) = I_1 \cap I_2 \neq I_1$ als auch $I(a_1) = I_1 \cap I_2 \neq I_2$. Zur Beschreibung von $I(a_1)$ sind beide Regeln erforderlich. In Abbildung 33c ist die Menge $I(a_1)$ als Schnittmenge aus I_1 und I_2 dargestellt.

Abbildung 34a stellt die auf diese Weise definierte Regelhierarchie auf Basis der abgeleiteten Dominator-Relationen zwischen den Regeltypen aus Θ_2 dar.

Zuweisungsregeln mit Bezug auf zwei Aktivitäten

Zuweisungsregeln mit Bezug auf zwei Aktivitäten

Betrachten wir nun die Zuweisungsregeln, die zwei Aktivitäten betreffen. Diese wurden in Abschnitt 4.2.2 definiert und sind in der Menge Θ_3 zusammengefasst.

$$\Theta_3 = \{\textit{binding}(A_1,A_2), \textit{separate}(A_1,A_2), \textit{orgDistMulti}(A_1,A_2,RT)\}$$

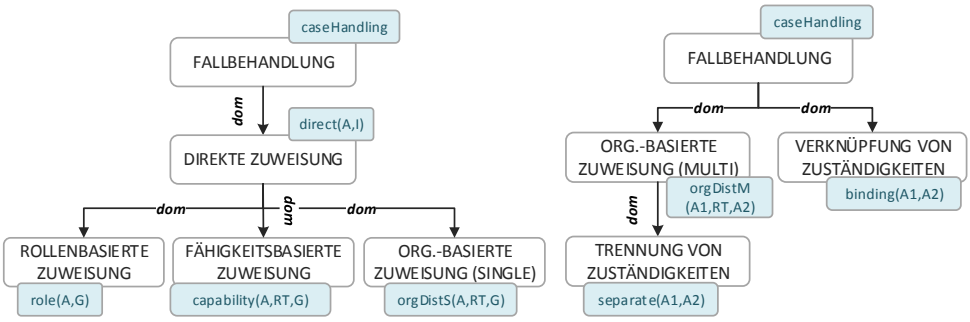
Ziel ist es wiederum eine Regelhierarchie der organisatorischen Zuweisungsregeln in Θ_3 aufzustellen. Wir vergleichen nun wiederum für alle Kombinationen der Regeltypen aus Θ_3 jeweils zwei Regeln r_1 und r_2, die für zwei beliebige Aktivitäten a_1 und a_2 abgeleitet wurden.

- ***separate***$(a_1,a_2) \not\rightarrow_{dom}$ ***binding***(a_1,a_2): Die *separate*-Regel besagt, dass a_1 und a_2 in Φ von verschiedenen Identitäten durchgeführt wurden. Die *binding*-Regel besagt hingehen, dass a_1 und a_2 in Φ von den gleichen Identitäten durchgeführt wurden. Im Fall dessen, dass beide Regeln abgeleitet werden würden, wäre $I(a_1) = I(a_2) = \emptyset$. Diese Regeln widersprechen sich demnach und werden nie gleichzeitig abgeleitet.

Irreflexive Relationen

Bei Betrachtung einer Regel *orgDistMulti*(a_1,a_2,rt_1), d.h. einer organisationsbasierten Zuweisungsregel die zwei Aktivitäten einbezieht, muss die Reflexivität der Relation rt_1 in dem gegebenen Organisationsmodell betrachtet werden. Wir gehen in dieser Arbeit davon aus, dass sämtliche Relationen in gegebenen Organisationsmodellen **irreflexiv** sind. Eine Relation rt_1 heißt irreflexiv, wenn die Beziehung $i\ rt_1\ i$ für keine Identität i des gegebenen Organisationsmodells gilt, d.h. also keine Identität in Relation zu sich selbst steht. Die *supervisor*-Relation ist beispielsweise irreflexiv, da eine Identität nicht die vorgesetzte Person von sich selbst sein kann. Sind die betrachteten Relationen des Organisationsmodells nicht irreflexiv, so befinden sich alle Regeltypen aus Θ_3 auf einer Hierarchieebene.

- ***orgDistMulti***$(a_1,a_2,rt_1) \not\rightarrow_{dom}$ ***binding***(a_1,a_2): Eine abgeleitete *orgDistMulti*(a_1,a_2,rt_1)-Regel besagt, dass eine Identität i_1 aus $I(a_1)$ in einer Relation rt_1 mit einer Identität i_2 aus $I(a_2)$ steht. Ausgehend davon, dass sämtliche Relationen rt im gegebenen Organisationsmodell irreflexiv sind, gilt dabei stets $i_1 \neq i_2$. Wie zuvor bereits erläutert, besagt eine abgeleitete *binding*-Regel hingehen, dass a_1 und a_2 in Φ von der gleichen Identität durchgeführt wurden, d.h. $i_1 = i_2$. Dies stellt einen Widerspruch dar und zeigt, dass diese beiden Regeln nie gleichzeitig abgeleitet werden.
- ***orgDistMulti***$(a_1,a_2,rt_1) \rightarrow_{dom}$ ***separate***(a_1,a_2): Die *orgDistMulti*-Regel ist spezieller als die *separate*-Regel. Im Fall dessen, dass beide Regeln für die Aktivitäten a_1 und a_2 abgeleitet werden, kann die *separate*-Regel entfernt werden. Betrachten wir hierzu die Menge der Identitäten $I(a_2)$ für eine spezielle Identität $i_1 \in I(a_1)$. Die abgeleitete *separate*-Regel besagt,



(a) Zuweisungsregeln mit Bezug auf eine Aktivität **(b)** Bezug auf zwei Aktivitäten

Abb. 34: Regelhierarchien organisatorischer Zuweisungsregeln

dass a_1 und a_2 von verschiedenen Identitäten durchgeführt werden, d.h. $i_1 \notin I(a_2)$. Ausgehend davon, dass sämtliche Relationen rt im gegebenen Organisationsmodell irreflexiv sind, besagt auch die *orgDistMulti*-Regel, dass $i_1 \notin I(a_2)$ gilt. Die Menge $I(a_2)$ ist im Fall der *orgDistMulit*-Regel jedoch kleiner als im Fall der *separate*-Regel, d.h. $I(a_2)_{org} \subset I(a_2)_{sep}$. Dieser Zusammenhang ist in Abbildung 33d beispielhaft visualisiert.

Dominator-Relationen für organisatorische Regeltypen

Abbildung 34b stellt die auf diese Weise definierte Regelhierarchie auf Basis der abgeleiteten Dominator-Relationen zwischen den Regeltypen aus Θ_3 dar. Die Regelhierarchien aus Abbildung 34 können nun dazu verwendet werden abgeleitete Modelle zu vereinfachen.

6.2.3 Perspektivenübergreifende Regeln

Regelhierarchie für perspektivenübergreifende Regeln

In diesem Abschnitt betrachten wir Regelhierarchien für **perspektivenübergreifende** Regelvorlagen am Beispiel der in Abschnitt 4.2.3 definierten perspektivenübergreifenden *sequence*-Regeln. Diese Regeltypen sind in der Menge Θ_4 zusammengefasst.

$$\Theta_4 = \{\textit{sequence}(A_1,A_2), \textit{roleSequence}(A_1,A_2,G), \textit{resourceSequence}(A_1,A_2,I)\}$$

Eine perspektivenübergreifende Regel r stellt gewisse Anforderungen bzw. Einschränkungen an die Ausführungsreihenfolge von Aktivitäten, gilt jedoch nur für eine bestimmte Menge an Identitäten $I(r)$, welche die Regel betrifft. Werden zwei Regeln r_1 und r_2 abgeleitet, welche die gleichen Anforderungen an die Ausführungsreihenfolge ausdrücken und $I(r_2) \subset I(r_1)$, so gilt $r_1 \rightarrow_{dom} r_2$. Betrachten wir hierzu ein Modell in dem sowohl eine *sequence*(a_1,a_2)- als auch eine *roleSequence*(a_1,a_2,g_1)-Regel enthalten ist. Da die *sequence*-Regel für alle definierten Identitäten gilt und die *roleSequence*-Regel nur für alle Identitäten in der Rolle g_1 gilt $I(\textit{roleSequence}) \subset I(\textit{sequence})$. Wenn für zwei Aktivitäten bereits unabhängig von der ausführenden Identität eine temporale Reihenfolge abgeleitet wurde (*sequence*), dann gilt diese temporale Reihenfolge

auch für bestimmte Teilmengen an Identitäten (*roleSequence*). Die *roleSequence*-Regel ist also in der *sequence*-Regel enthalten und kann daher entfernt werden, d.h. *sequence*(a_1,a_2) $\rightarrow_{dom}$ *roleSequence*(a_1,a_2,g_1). Der gleiche Zusammenhang gilt identisch für die *resourceSequence*-Regel, d.h. *roleSequence*(a_1,a_2,g_1) $\rightarrow_{dom}$ *resourceSequence*(a_1,a_2,i_1), da I(*resourceSequence*) $\subset$ I(*roleSequence*).

6.2.4 Einordnung benutzerdefinierter Regeltypen

Einordnung benutzerdefinierter Regelvorlagen

In Abschnitt 4.3 wurde anhand von *bindingRole*-Regeln gezeigt, wie der Suchraum durch Definition neuer Regelvorlagen erweitert werden kann. Benutzerdefinierte Regelvorlagen können in existierende Regelhierarchien eingeordnet werden. Eine *bindingRole*-Regel besagt beispielsweise, dass zwei Aktivitäten a_1 und a_2 von Identitäten in der gleichen Rolle durchgeführt werden. Enthält das abgeleitete Modell auch die stärkere *binding*-Regel zwischen den Aktivitäten a_1 und a_2, so ist die *bindingRole*-Regel redundant und kann entfernt werden. Die Regelhierarchie in Abbildung 34b wird entsprechend durch eine weitere Relation *binding* $\rightarrow_{dom}$ *bindingRole* erweitert. Der gleiche Zusammenhang würde beispielsweise für *separate*- und *separateRole*-Regeln gelten, d.h. die Regelhierarchie würde durch die Relation *separate* $\rightarrow_{dom}$ *separateRole* erweitert.

6.2.5 Berücksichtigung unterschiedlicher Modalitäten

Kein Pruning bei Regeln unterschiedlicher Modalität

In Abschnitt 4.1.4 wurde gezeigt, dass abgeleitete DPIL-Prozessmodelle sowohl verpflichtende als auch lediglich empfohlene Regeln enthalten können. Es stellt sich daher die Frage, ob die Möglichkeit zur Entfernung von Regeln von deren Modalität abhängt. Betrachten wir hierzu als Beispiel die beiden abgeleiteten Regeln *advice direct*(a_1,i_1) und *ensure role*(a_1,g_1). Die beiden Regel besagen, dass Aktivität a_1 stets von einer Identität in der Rolle g_1 durchgeführt wird und bevorzugt von der konkreten Identität i_1. Würde man die schwächere jedoch verpflichtende *role*-Regel entfernen, würde die Aussage des Modells verändert. a_1 könnte jetzt von allen möglichen Identitäten durchgeführt werden. Dieses Beispiel zeigt, dass die Entfernung von Regeln auf Basis von Regelhierarchien nur dann möglich ist, wenn die betrachteten Regeln verpflichtend sind. Sämtliche vorgestellten Konzepte zur Entfernung von Regeln auf Basis von Regelhierarchien werden nun an einem Beispiel erläutert.

BEISPIEL (VEREINFACHUNG AUF BASIS VON REGELHIERARCHIEN) Betrachten wir den Regelteil eines erzeugten Beispiel-Prozessmodells in Listing 10. Das abgeleitete Modell besagt, dass Aktivität a_1 stets von einer Identität in der Rolle g_1 bearbeitet wird, jedoch vorzugsweise von der konkreten Identität i_1. Die Aktivitäten a_2 und a_3 werden stets von der gleichen Identität bearbeitet. a_1 muss vor a_2 und a_2 vor a_3 durchgeführt werden, wobei a_2 vorzugsweise direkt vor a_3 durchgeführt wird. Die *roleSequence*-Regeln zwischen a_1 und a_2 können entfernt werden, da diese bereits in der stärkeren *sequence*-Regel enthal-

ten sind. Das gleiche gilt auch für die *bindingRole*-Regel zwischen a_2 und a_3, welche schwächer als die *binding*-Regel ist. Das Entfernen der *role*-Regel von a_1 ist trotz der ebenfalls abgeleiteten *direct*-Regel nicht möglich, da diese lediglich empfohlen ist. Ein Entfernen der *sequence*(a_2,a_3)-Regel ist nicht möglich, da auch hier die Aussage des Modells verändert werden würde. Beide Regeln sind notwendig um die extrahierten Informationen lückenlos darzustellen. Die *sequence*-Regel liefert die Information, dass a_2 auf jeden Fall vor a_3 durchgeführt werden muss. Die *directSequence*-Regel besagt zusätzlich, dass a_2 direkt vor a_3 durchgeführt werden sollte, was jedoch lediglich eine Empfehlung darstellt. Ohne die *sequence*-Regel wäre die Ausführungsreihenfolge von a_2 und a_3 nicht mehr sichergestellt. Im Modell 11 sind redundante Regeln entfernt.

Listing 10: Modell (zuvor)

```
process Example {
   ...
   ensure role(a1,g1)
   advice direct(a1,i1)
   ensure binding(a2,a3)
   ensure bindingRole(a2,a3)
   ensure sequence(a1,a2)
   ensure roleSequence(a1,a2,g1)
   ensure roleSequence(a1,a2,g2)
   ensure sequence(a2,a3)
   advice directSequence(a2,a3)
}
```

Listing 11: Modell (reduziert)

```
process Example_Pruned {
   ...
   ensure role(a1,g1)
   advice direct(a1,i1)
   ensure binding(a2,a3)

   ensure sequence(a1,a2)



   ensure sequence(a2,a3)
   advice directSequence(a2,a3)
}
```

6.3 TRANSITIVE REDUKTION

Transitiv ableitbare Regeln

Erzeugte DPIL-Prozessmodelle können auch durch das Entfernen von **transitiv ableitbaren Regeln** vereinfacht werden. Hierzu betrachten wir Regelvorlagen, die zwei Aktivitäten betreffen. Ein Regeltyp R ist **transitiv**, wenn aus den Regelinstanzen $r_1(a_1, a_2)$ und $r_2(a_2, a_3)$ auch $r_3(a_1, a_3)$ folgt. Die erzeugten Regeln eines transitiven Regeltyps bilden einen gerichteten Graphen. Durch die Analyse dieser Graphen können redundante, transitiv ableitbare Regeln identifiziert und entfernt werden. Dieser Vorgang wird als **transitive Reduktion** bezeichnet. Transitive Reduktion kann auch bei einer Kombination aus stärkeren und schwächeren Regeln einer Regelhierachie angewendet werden. Um bei der Modell-Nachbearbeitung auch benutzerdefinierte Regelvorlagen einzubeziehen, muss für jede benutzerdefinierte Vorlage angegeben werden, ob diese transitiv reduzierbar ist.

Transitive Reduktion von Modellen

6.3.1 Verhaltensorientierte Perspektive

Transitive verhaltensorientierte Regeltypen

Betrachten wir zuerst transitive Regeltypen der **verhaltensorientierten Perspektive**, wie z.B. *directSequence*- und *sequence*-Regeln. Betrachten wir hierzu das Prozessmodell in Listing 12, das *sequence*- bzw. *directSequence*-Regeln enthält. Die *sequence*-Regel zwischen a_1 und a_3 kann transitiv aus der *directSequence*-Regel zwischen a_1 und a_2 und der *sequence*-Regel zwischen a_2 und a_3 gefolgert werden. Die *directSequence*-Regel impliziert die schwächere *sequence*-Regel, wodurch die transitive Reduktion ermöglicht wird und die *sequence*-Regel zwischen a_1 und a_3 entfernt werden kann. Zur transitiven Reduktion von erzeugten Prozessmodellen werden daher verschiedene Mengen abgeleiteter Regeln für jede Regelhierarchie gebildet, welche jeweils einen gerichteten Graphen darstellen. Anschließend wird jede Menge, d.h. jeder Teilgraph, durch Anwendung des Algorithmus von Aho et al. [16] transitiv reduziert. Die resultierenden Regelmengen werden wieder zur Menge an Prozessregeln R des DPIL-Prozessmodells vereint. Das Ergebnis ist im Modell in Listing 13 ersichtlich.

Listing 12: Modell (zuvor)

```
process Example {
   ...
   ensure directSequence(a1,a2)
   ensure sequence(a1,a3)
   ensure sequence(a2,a3)
}
```

Listing 13: Modell (reduziert)

```
process Example_Reduced {
   ...
   ensure directSequence(a1,a2)

   ensure sequence(a2,a3)
}
```

6.3.2 Organisatorische Perspektive

Transitive organisatorische Regeltypen

Neben verhaltensorientierten Regeln können auch Regeln der **organisatorischen Perspektive** teilweise transitiv reduziert werden. Bei der transitiven Reduktion müssen, wie bereits erläutert, nur Regeltypen betrachtet werden, die mehrere Aktivitäten einbeziehen, d.h. die Regeltypen in Θ_3.

- ***binding***-Regeln sind reduzierbar. Aus *binding*(a_1,a_2) und *binding*(a_2,a_3) folgt *binding*(a_1,a_3). Müssen die Aktivitäten a_1 und a_2 sowie die Aktvitäten a_2 und a_3 von der gleichen Person bearbeitet werden, dann gilt dies auch für a_1 und a_3. *Binding*-Regeln in erzeugten Modellen lassen sich daher transitiv reduzieren.

- ***separate***-Regeln sind *nicht transitiv* reduzierbar, d.h. wird Aktivität a_1 in Φ nicht von der gleichen Identität durchgeführt wie a_2 und a_2 wird nicht von der gleichen Identität durchgeführt wie a_3, so lässt sich nicht automatisch folgern, dass auch a_1 und a_3 von unterschiedlichen Identitäten durchgeführt werden. Dies lässt sich durch ein einfaches Gegenbeispiel belegen: Angenommen die Aktivitäten a_1 und a_3 werden immer von i_1 bearbeitet und a_2 wird immer von i_2 bearbeitet. Dann gelten die Regeln

separate(a_1,a_2) und *separate*(a_2,a_3) jedoch nicht *separate*(a_1,a_3). *Separate*-Regeln sind daher nicht allgemeingültig transitiv reduzierbar.

- ***orgDistMulti***-Regeln sind *transitiv* reduzierbar, wenn die betrachteten Regeln den gleichen organisatorischen Relationstyp betreffen und es sich dabei um einen transitiven Relationstyp handelt. Enthält ein abgeleitetes Modell beispielsweise die Regeln *orgDistM*(a_1,a_2,rt_1), *orgDistM*(a_2,a_3,rt_1) und *orgDistM*(a_1,a_3,rt_1), wobei rt_1 einen transitiven Relationstyp darstellt, so ist die letztgenannte Regel redundant und lässt sich aus den anderen beiden Regeln folgern. Ein Beispiel eines transitiven Relationstyps ist die *supervisor*-Relation.

6.3.3 Perspektivenübergreifende Regeln

Transitivität bei perspektivenübergreifenden Regeln

Die **perspektivenübergreifenden Regeln** *roleSequence* und *resourceSequence* sind *transitiv* reduzierbar unter der Bedingung, dass die betrachteten Regeln die gleiche Menge an Identitäten $I(r)$ betreffen. Beispielsweise sind nur *roleSequence*-Regeln reduzierbar, welche die gleiche organisatorische Gruppe betreffen. Nur diese Regeln sind jeweils Teil des gleichen gerichteten Graphen. Das Modell in Listing 14 kann daher transitiv reduziert werden, wohingegen im Modell in Listing 15 eine Regel nicht Teil des betrachteten Graphen ist was eine transitive Reduktion des Modells verhindert.

Listing 14: Modell (reduzierbar)

```
process Example_Reducable {
   ...
   ensure roleSequence(a1,a2,g1)
   ensure roleSequence(a2,a3,g1)
   ensure roleSequence(a1,a3,g1)
}
```

Listing 15: Modell (nicht reduzierbar)

```
process Example_NotReducable {
   ...
   ensure roleSequence(a1,a2,g1)
   ensure roleSequence(a2,a3,g2)
   ensure roleSequence(a1,a3,g1)
}
```

6.3.4 Berücksichtigung unterschiedlicher Modalitäten

Keine transitive Reduktion bei Empfehlungen

Abgeleitete DPIL-Prozessmodelle können sowohl verpflichtende (*ensure*) als auch lediglich empfohlene Regeln (*advice*) enthalten. Es stellt sich daher die Frage, ob die Möglichkeit zur Entfernung von Regeln von deren Modalität abhängt. Die transitive Reduktion von erzeugten Modellen ist nur im Fall von verpflichtenden Regeln möglich. Sind neben verpflichtenden auch empfohlene Regeln in dem betrachteten Modell enthalten, stellt ein transitiv reduziertes Modell nicht mehr den ursprünglich abgeleiteten Prozess dar. Wir belegen diese These durch ein einfaches Beispiel. Betrachten wir hierzu die beiden Modelle in Listing 16 und Listing 17.

Listing 16: Modell (zuvor)

```
process Example_BeforeReducing {
   ...
   ensure directSequence(a1,a2)
   advice sequence(a2,a3)
   ensure sequence(a1,a3)
}
```

Listing 17: Modell (danach)

```
process Example_AfterReducing {
   ...
   ensure directSequence(a1,a2)
   advice sequence(a2,a3)

}
```

Im Modell in Listing 16 sind die *sequence*-Regeln zwischen a_1 und a_2 sowie zwischen a_1 und a_3 verpflichtend, wohingegen die *sequence*(a_2,a_3)-Regel lediglich empfohlen ist. Das Modell in Listing 17 stellt das transitiv reduzierte Modell dar, in dem die *sequence*-Regel zwischen a_1 und a_3 entfernt wurde. Voraussetzung für die transitive Reduktion eines Modells ist, dass sich dich die Semantik, d.h. die Bedeutung, des Modells durch die Reduktion nicht verändert. Im Modell links **muss** a_1 stets beendet sein bevor mit a_3 begonnen werden kann. Das Modell rechts trifft eine andere Aussage über den Prozess. Hier kann a_3 auch gestartet werden, ohne dass a_1 zuvor beendet wurde. Die Semantik des ursprünglichen Modells wird durch die transitive Reduktion also verändert. Eine transitive Reduktion ist im Fall von nicht verpflichtenden Regeln somit nicht möglich.

6.4 ZUSAMMENFASSUNG

In diesem Kapitel wurde betrachtet, wie durch **Nachbearbeitung** die Komplexität erzeugter DPIL-Modelle verringert und dadurch die Lesbarkeit erleichtert werden kann. Zur **Steigerung der Verständlichkeit** extrahierter Modelle wird die Anzahl an notwendigen Regeln zur Beschreibung des Prozesses möglichst minimiert.

Hierzu können zwei Methoden zur Reduktion bzw. **Entfernung redundanter Regeln** angewendet werden. Einerseits können Regeln auf Basis von gegebenen Regelhierarchien entfernt werden. Bestimmte Regeln können aus dem betrachteten Modell entfernt werden, wenn diese bereits in anderen, stärkeren Regeln enthalten sind. Das Entfernen von Regeln auf Basis von Regelhierarchien ist nur dann möglich, wenn die betrachteten Regeln verpflichtend ist. Die zweite Methode zur Entfernung redundanter Regeln basiert auf der Transitivität bestimmter Regeltypen. Abgeleitete Regeln eines transitiven Regeltyps, wie z.B. *sequence*-Regeln, bilden einen gerichteten Graphen. In diesem Graph können redundante, transitiv ableitbare Regeln identifiziert und entfernt werden. Auch die transitive Reduktion von erzeugten Modellen ist nur im Fall von verpflichtenden Regeln möglich. Die Verbesserung der Lesbarkeit durch die Nachbearbeitung von Modellen wird im folgenden Kapitel bei der Anwendung auf reale Ereignisprotokolle ersichtlich.

7 EVALUATION

INHALT

In diesem Kapitel wird die prototypische Implementierung des vorgestellten Ansatzes beschrieben. Des Weiteren werden die Analysefähigkeiten des vorgestellten Process Mining-Ansatzes durch Anwendung auf zwei Ereignisprotokolle exemplarischer agilen, personenbezogener Prozesse evaluiert.

7.1 TOOL-UNTERSTÜTZUNG DURCH DEN DPILMINER

Prototypische Umsetzung durch DpilMiner

Drools.NET

Der vorgestellte Ansatz wird durch die Microsoft .NET C#-Applikation *DpilMiner* umgesetzt. Das System besteht dabei aus vier verschiedenen Komponenten, die dem konzeptuellen Aufbau des vorliegenden Ansatzes entsprechen (vgl. Abbildung 37). Das System führt dabei Schrittweise durch den Analyseprozess. Zur Implementierung des Rete-Algorithmus wird die .NET-Portierung der Drools-Plattform (*Drools.NET*) verwendet.

Ereignisprotokoll Import

MODUL 1: EREIGNISPROTOKOLL IMPORTER Der erste Schritt zur Analyse eines Ereignisprotokolls mit dem DpilMiner ist das Einlesen bzw. Importieren einer Logdatei. Das System unterstützt das Importieren von XML-Logdateien nach dem *XES-Standard* (Abschnitt 2.3.4). Die zu analysierende XES-Datei kann dabei über ein Dialog-Fenster ausgewählt werden. Um auch große Logdateien importieren zu können, wird der Fortschritt des Importvorgangs stetig visualisiert.

Konfiguration und Voranalyse

MODUL 2: KONFIGURATION UND VORVERARBEITUNG Nach dem Einlesen des Ereignisprotokolls können die zu analysierenden Regelvorlagen aus einer Liste ausgewählt werden. Im DpilMiner sind standardmäßig sämtliche Regelvorlagen der DPIL-Standardmodellierungsbibliothek, weitere Muster der Workflow Resource Patterns (*organisationsbasierte Zuweisung*, *fähigkeitsbasierte Zuweisung*) sowie perspektivenübergreifende Regeltypen (*roleSequence*, *resourceSequence*) abgebildet. Durch die Auswahl der entsprechenden Regelvorlagen wird der Analysevorgang konfiguriert. Auf Basis der zur Analyse ausgewählten Regelvorlagen wird das Ereignisprotokoll, wie in Abschnitt 5.4.1 beschrieben, zu verschiedenen Transaktionsdatenbanken transformiert. Das System enthält eine Implementierung des Apriori-Algorithmus zur Ableitung von häufig auftretenden Parameterkombinationen aus diesen Transaktionsdatenbanken. Der Nutzer spezifiziert hierfür den Support-Schwellenwert, den eine Parameterkombinationen mindestens erfüllen muss. Als Ergebnis liegen die entsprechenden L1- bzw. L2-Itemsets der zur Instanziierung der Regelvorlagen benötigten häufigen Parameterkombinationen vor. Die Benutzeroberfläche des Vorverarbeitungsmoduls des DpilMiners ist in Abbildung 35 dargestellt.

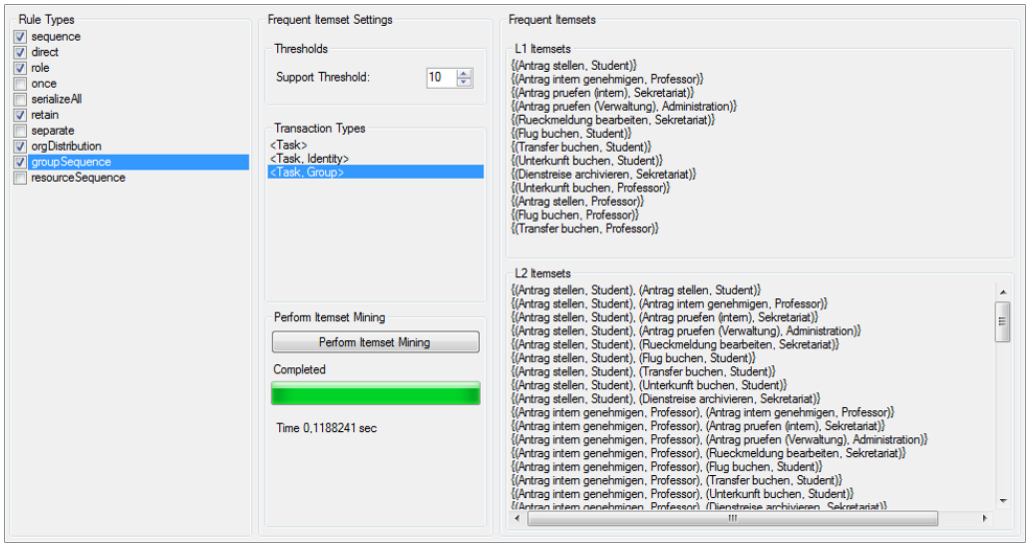


Abb. 35: Vorverarbeitungsmodul des DpilMiners

Mining-Modul

MODUL 3: MINING Auf Basis der extrahierten häufigen Parameterkombinationen werden die ausgewählten Regelvorlagen instanziiert und die zu überprüfenden Regelkandidaten erzeugt. Anhand dieser Menge an Regeln wird das Rete-Netzwerk (Abschnitt 4.5.2) **einmalig** aufgebaut. Der eigentliche Analyse-Vorgang kann sowohl seriell (*Single-Threading*) als auch parallel (*Multi-Threading*) ausgeführt werden. Im Fall der seriellen Regelüberprüfung werden die Spuren des betrachteten Logs sequentiell eingelesen. Dieses Verfahren ist zwar langsamer, erfordert jedoch wesentlich weniger Arbeitsspeicher, da lediglich die Informationen der aktuell betrachteten Prozessinstanz im Speicher gehalten werden müssen. Der folgende

Serielle Regelüberprüfung

C#-Code zeigt einen Ausschnitt der Implementierung der seriellen Regelüberprüfung.

```
List<Event> eventsOfCurrentInstance = reader.readXMLInstanceEvents();

while (eventsOfCurrentInstance != null)
{
        workingMemory = ruleBase.NewWorkingMemory();
        ...
        foreach (Event ev in eventsOfCurrentInstance)
                workingMemory.assertObject(ev);
        ...
        workingMemory.fireAllRules();
        eventsOfCurrentInstance = reader.readXMLInstanceEvents();
}
reader.InstanceReader.Close();
```

Zu Beginn werden die Ereignisinformationen der aktuellen Instanz durch den XES-Importer eingelesen. Das *Working Memory* wird für jede Instanz neu initialisiert und die Ereignisse der betracheten Instanz hinzugefügt. Durch die *fireAllRules*-Methode werden sämtliche Regelkandidaten im aktuellen Working Memory überprüft. Im Fall des parallelen Analyse-Vorgangs werden sämtliche Spuren des Protokolls auf einmal in den Arbeitsspeicher eingelesen. Je nach zur Verfügung stehender Hardware, d.h. Prozessoranzahl, verteilt die .NET-Laufzeitumgebung die Überprüfung der konkreten Instanzen durch Erzeugung von Threads auf die verschiedenen Prozessorkerne. Hierfür steht im .NET-Framework die Klasse *Parallel* zur Verfügung. Der folgende C#-Code zeigt einen Ausschnitt der Implementierung der parallelisierten Regelüberprüfung.

Parallele Regelüberprüfung

```
List<Instance> instances = reader.readXMLComplete();

Parallel.ForEach(instances, currentInstance =>
{
        WorkingMemory workingMemory = ruleBase.NewWorkingMemory();
        ...
        foreach (Event ev in currentInstance.Events)
                workingMemory.assertObject(ev);
        ...
        workingMemory.fireAllRules();
}
```

Sämtliche Instanzen liegen nach dem Einlesen der vollständigen XES-Datei in einer Liste (*instances*) vor. Die Verteilung und Thread-Erzeugung erfolgt anschließend automatisiert durch das .NET-Framework. Der bearbeitende Thread erzeugt sich zur Überprüfung der zugeordneten Instanz ein eigenes *WorkingMemory*. Diesem werden anschließend sämtliche Ereignisse der aktuell betrachteten Instanz hinzugefügt. Im letzten Schritt werden sämtliche Regelkandidaten für diese Instanz überprüft (*fireAllRules*).

Nach der Regelüberprüfung liegt für jeden Regelkandidaten die Anzahl an Instanzen vor, in denen die Regel nicht-trivial erfüllt ist. Anschließend werden die Regelkandidaten, wie in den Abschnitten 4.1.3 und 4.1.4 beschrieben, durch den benutzerdefinierten Support-Wert gefiltert und durch die benutzerdefinierten Confidence-Werte klassifiziert. Als Resultat zeigt das System die als verpflichtend und als empfohlen klassifizierten Regeln, sowie deren jeweiligen Support- und Confidence-Wert. Die Benutzeroberfläche des Mining-Moduls des DpilMiners ist in Abbildung 36 dargestellt.

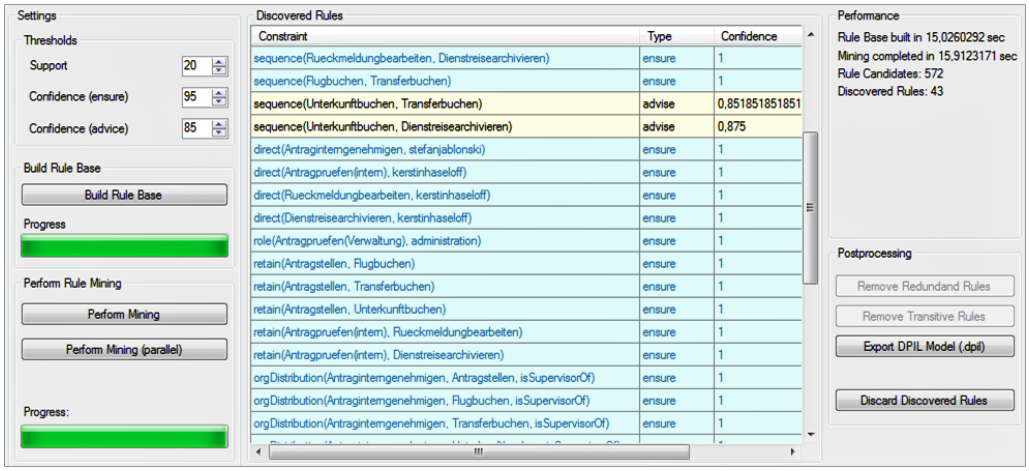


Abb. 36: Analyse- bzw. Mining-Modul des DpilMiners mit exemplarischen, extrahierten Regeln

Modell-Nachbearbeitung und Export

MODUL 4: MODELL-NACHBEARBEITUNG UND EXPORT Die Möglichkeit zur Nachbearbeitung der abgeleiteten Regeln befindet sich im rechten unteren Teil von Abbildung 36. Dieses Modul bietet die Möglichkeit die Menge an abgeleiteten Regeln auf Redundanzen hin zu untersuchen und diese bei Bedarf zu entfernen. Einerseits können schwächere Regeln entfernt werden, die bereits in anderen stärkeren Regeln enthalten sind (*Remove Redundant Rules*). Im DpilMiner sind hierzu exemplarisch die Regelhierarchien der abgebildeten Regelvorlagen aus Abschnitt 6.2 hinterlegt. Des Weiteren können transitiv automatisch ableitbare Regeln (Abschnitt 6.3) entfernt werden (*Remove Transitive Rules*). Im Anschluss kann das vollständige, ausführbare DPIL-Prozessmodell (Struktureller Teil und Regelteil) als *.dpil-Datei* erzeugt und abgespeichert werden. Dieses Modell kann in entsprechenden manuellen Modellierungswerkzeugen, wie der ModelWorkbench [108] analysiert und nachbearbeitet werden. Anschließend kann das Modell auf der ProcessNavigation-Plattform[141] ausgeführt werden.

Vollständiger Aufbau des DpilMiners

Abbildung 37 gibt einen Überblick über den vollständigen Aufbau und das Zusammenspiel der einzelnen Module des DpilMiners. Die manuell justierbaren Stellschrauben des Analyseprozesses $\mathtt{minSupp}$ (Vorverabeitung), ausgewählte Regelvorlagen, $\mathtt{minSupp}$ (Regeln), $\mathtt{minConf}_S$, $\mathtt{minConf}_H$ sind dabei

hervorgehoben. Die Dokumentensymbole zeigen die Datenflüsse zwischen den einzelnen Modulen.

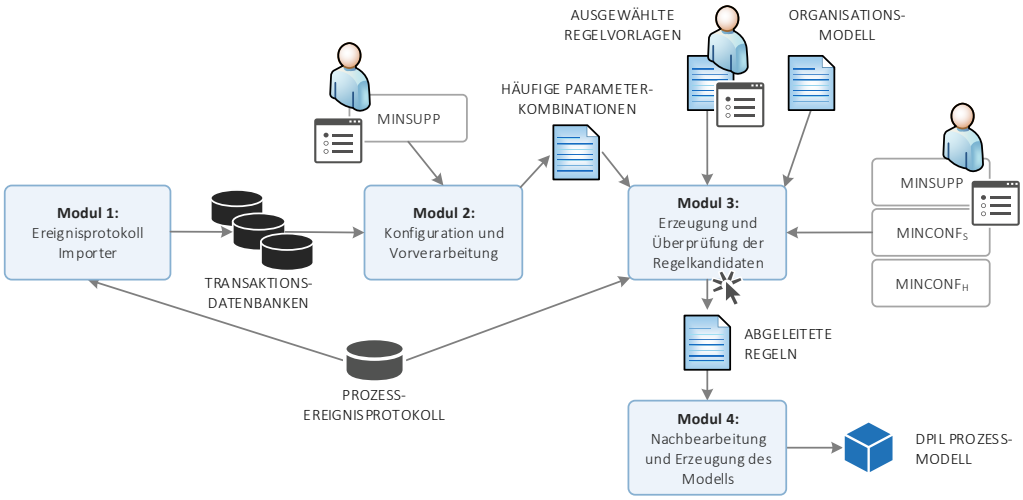


Abb. 37: Vollständiger Aufbau und Module des DpilMiners

Nach der Beschreibung des Analyse-Werkzeugs, wird der vorliegende Ansatz in den folgenden Abschnitten durch die Analyse exemplarischer Ereignisprotokolle mit dem DpilMiner evaluiert und die Ergebnisse interpretiert.

7.2 ANALYSE EINES DIENSTREISEPROZESSES

Zur praktischen Evaluation des vorliegenden Ansatzes wird zuerst die Anwendung auf ein Ereignisprotokoll eines universitären Dienstreiseabwicklungsprozesses beschrieben.

Agiler Dienstreiseprozess

Der Prozess ist, anders als bei einem Verwaltungsprozess zunächst vermutet werden könnte, eher agil als strikt. Prozessbeteiligte Personen weichen häufig von der vorgegeben Reihenfolge an Aktivitäten ab, was zu vielen unterschiedlichen Ausführungsvarianten führt. Zudem ist die Durchführung des Prozesses abhängig von einer Vielzahl an organisatorischen Zusammenhängen. Der Dienstreiseabwicklungsprozess stellt daher ein adäquates Anwendungsbeispiel für das vorliegende Verfahren dar. Da der vollständige Prozess als agil eingestuft werden kann und keine strikten Teilprozesse enthält, ist keine Aufteilung des Ereignisprotokolls in strikte und agile Teile, wie in Abschnitt 5.3 beschrieben, erforderlich. Das Ereignisprotokoll enthält 2104 Ereignisse (Start und Abschluss-Ereignisse) die 10 verschiedene Aktivitäten betreffen. Das Start-Ereignis der Aktivität "Antrag stellen" zur Dienstreiseabwicklung der Person "SS" nach Riga/Lettland wurde beispielsweise wie folgt im XML-basierten XES-Format aufgezeichnet:

Keine Aufteilung des Logs erforderlich

```
<string key="concept:name" value="SS_Riga2013"/>
<event>
        <string key="org:resource" value="SS"/>
```

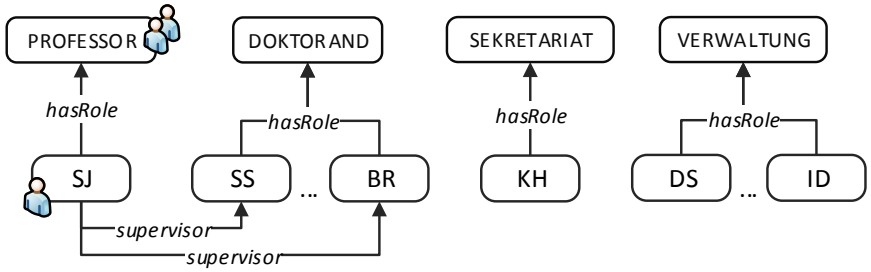


Abb. 38: Organisationsmodell einer Forschungsgruppe

```
        <date key="time:timestamp" value="2013-11-06T14:58:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
```

Zwei Spuren, d.h. die Ereignisse von zwei vollständigen Prozessinstanzen dieses Prozesses, sind in Anhang A.1.1 zu finden. Der Dienstreiseabwicklungsprozess wurde für 6 Monate unter Beteiligung von 10 verschiedenen Personen einer universitären Forschungsgruppe aufgezeichnet. Die beteiligten Personen verwendeten während der Durchführung einer Dienstreiseabwicklung das in Abschnitt 3.3.4 beschriebene *ProcessNavigation*-System. Das System wurde dabei mit einem DPIL-Modell konfiguriert, in dem lediglich die Aktivitäten des Prozesses definiert sind, jedoch keinerlei einschränkende Regeln (vgl. Abschnitt 3.3.5). Benutzer hatten deshalb uneingeschränkte Flexibilität während der Ausführung des Prozesses, so dass alle möglichen Varianten protokolliert werden konnten. Auf diese Weise wurden insgesamt 128 verschiedene Dienstreise-Fälle protokolliert.

Aus Tabelle 8 geht hervor, dass bestimmte organisatorische Zusammenhänge nur dann entdeckt werden können, wenn das zu Grunde liegende Organisationsmodell vorliegt. Um die **vollständige Funktionalität** des Ansatzes aufzeigen zu können, wird daher neben dem Ereignisprotokoll auch das Organisationsmodell der betrachteten Domäne benötigt. Das gegebene Organisationsmodell der Forschungsgruppe, welches in Abbildung 38 dargestellt ist, ordnet den 10 prozessbeteiligten Personen 4 verschiedene Rollen zu, d.h. Doktorand, Professor, Mitarbeiterinnen des Sekretariats sowie Mitarbeiterinnen und Mitarbeiter der Verwaltung. Des Weiteren enthält das Organisationsmodell die Vorgesetzten-Relationen (*RelationType supervisor*) von Professoren und Doktoranden, von denen exemplarisch die Identitäten "SS" und "BR" visualisiert sind.

Organisationsmodell der betrachteten Domäne

7.2.1 Analyse durch den α-Algorithmus

Zuerst wird das gegebene Ereignisprotokoll mit einem klassischen, prozeduralen Analyseverfahren analysiert, dem in Kap. 2 beschriebenen α-Algorithmus [7]. Das resultierende Modell ist in Abbildung 39 dargestellt und im Anhang A.1.2 vollständig ersichtlich.

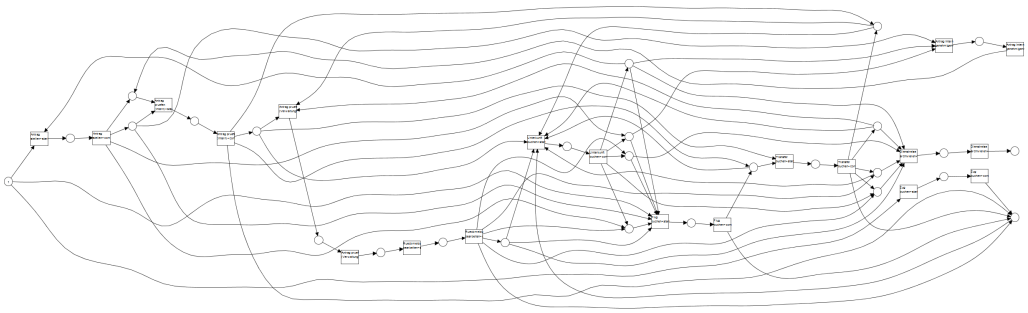


Abb. 39: Abgeleitetes Petrinetz des Dienstreiseverwaltungsprozesses durch den α-Algorithmus

Probleme des α-Algorithmus

Die Probleme und Schwächen des α-Algorithmus bei der Analyse dieses agilen Prozesses werden anhand des Modells ersichtlich. Das Modell enthält 20 verschiedene Knoten für die Start- und Beendigungsereignisse der 10 Aktivitäten und **82** verschiedene Kanten. Das Modell zeigt, dass der Prozess in bestimmten Instanzen mit Aktivitäten begonnen wurde, welche in anderen Instanzen erst gegen Ende des Prozess auftreten. Dies ist auf die Agilität des Prozesses und die gegebenen Freiräume der Benutzer zurückzuführen. Bei der Analyse mit dem α-Algorithmus führt diese Tatsache, trotz der relativ geringen Anzahl an Aktivitäten, zu einem tendenziell unüberschaubaren Modell, da zahlreiche explizite Sequenzflusslinien von der einen Seite zur anderen Seite des Modells (und zurück) erzeugt werden. In dem Modell ist des Weiteren jeder Pfad gleich bewertet, d.h. es wird nicht zwischen verpflichtenden und lediglich empfohlenen Aktionen unterschieden. Wie bereits beschrieben, erzeugt der α-Algorithmus ein Petrinetz, welches rein den Kontrollfluss des Prozesses beschreibt. Eine Analyse der organisatorischen Perspektive bzw. deren Einfluss auf den Kontrollfluss, d.h. von perspektivenübergreifenden Zusammenhängen, ist daher nicht möglich.

7.2.2 Abgeleitetes DPIL-Modell

Konfiguration des DpilMiners

In diesem Abschnitt wird das betrachtete Ereignisprotokoll nun mit dem *DpilMiner* und unterschiedlichen Konfigurationen analysiert. Zur vollständigen Analyse des Ereignisprotokolls werden zwei verschiedene Mengen an Regelvorlagen gewählt. Die Regelvorlagen der Menge Θ_1={*direct*, *role*, *binding*, *orgDistMulti*} analysieren Prozessmuster die rein die **organisatorische Perspektive** des Prozesses betreffen, d.h. die Zuweisung von Identitäten zu Aktivitäten. Die Regelvorlagen der Menge Θ_2={*sequence*, *roleSequence*} analysieren hingegen die Ausführungsreihenfolge der Aktivitäten (*sequence*) und unter Berücksichtigung der organisatorischen Perspektive (*roleSequence*), d.h. **verhaltensorientierte** und **perspektivenübergreifende Zusammenhänge**.

Widmen wir uns zunächst dem abgeleiteten Modell, d.h. den extrahierten Prozessregeln. Das vollständige extrahierte, nachbearbeitete Prozessmodell ist in Anhang A.1.3 visualisiert. Damit möglichst keine seltenen aber dennoch re-

levanten Zusammenhänge übersehen werden, wählen wir in der Vorverarbeitungsphase $minSupp = 10\%$. Um interessante von eher unwichtigen extrahierten Regeln zu trennen, wählen wir $minSupp = 20\%$ zur Filterung von Regeln. Zur Klassifikation von Kandidaten in verpflichtende und lediglich empfohlene Regeln wählen wir $minConf_H = 95\%$ und $minConf_S = 85\%$. Auf diese Weise wird gewährleistet, dass fehlerhafte Aufzeichnungen oder Ausnahmen die Entdeckung verpflichtender Regeln nicht verfälschen.

Abgeleitete organisatorische Zuweisungsregeln

Wir betrachten zuerst die extrahierten Regeln bei der Analyse mit Θ_1, d.h. rein **organisatorische Zusammenhänge**. Auf Basis des betrachteten Ereignisprotokolls und des Organisationsmodells aus Abbildung 38 werden mit dem DpilMiner insgesamt 14 verschiedene organisatorische Regeln extrahiert. Die folgenden Regeln in Listing 18 zeigen einen Ausschnitt des extrahierten Modells.

Listing 18: Abgeleitete Regeln der organisatorischen Perspektive

```
%Regeln der organisatorischen Perspektive
ensure direct(Antrag intern genehmigen, SJ)
ensure role(Antrag pruefen (Verwaltung), Administration)
ensure binding(Antrag stellen, Flug buchen)
ensure binding(Antrag stellen, Unterkunft buchen)
ensure binding(Antrag stellen, Transfer buchen)
ensure orgDistMulti(Antrag intern genehmigen, Antrag stellen, supervisor)
```

Die abgeleiteten Regeln zeigen, dass die Aktivität "Antrag intern genehmigen" immer von der Identität "SJ" durchgeführt wurde (*verpflichtende direkte Zuweisung*). Die Aktivität "Antrag prüfen (Verwaltung)" wurde stets von Identitäten in der Rolle "Administration" durchgeführt (*verpflichtende rollenbasierte Zuweisung*). Die drei *binding*-Regeln zeigen, dass die Buchung von Flügen, Unterkünften und Transfers vom Antragsteller durchzuführen sind (*verpflichtende Verknüpfung von Zuständigkeiten*). Das Modell zeigt zusätzlich, dass die Person, die einen Antrag intern genehmigt, tendenziell der Vorgesetzte des Antragstellers ist (*verplichtende organisatorische Zuweisung*).

Regeln der verhaltensorientierten Perspektive

Betrachten wir nun die extrahierten Regeln bei der Analyse mit der Menge an Regelvorlagen Θ_2, d.h. die **verhaltensorientierte Perspektive** und **perspektivenübergreifende Zusammenhänge**. Hier werden insgesamt 20 verschiedene Regeln abgeleitet. Betrachten wir zuerst die Analyse der Ausführungsreihenfolge von Aktivitäten, d.h. rein die **verhaltensorientierte Perspektive** des betrachteten Ereignisprotokolls. Zuvor wurde gezeigt, dass prozedurale Verfahren ein komplexes, schwer verständliches, flussorientiertes Prozessmodell erzeugen. Zur Betrachtung der verhaltensorientieren Perspektive analysieren wir das Ereignisprotokoll auf Basis der *sequence*-Regelvorlage. Anstatt der 82 Sequenzflusslinien im abgeleiteten Modell des α-Algorithmus, wird das Verhalten des Prozesses im resultierenden Modell des DpilMiners mit 14 *sequence*-Regeln (6 verpflichtende und 8 empfohlene Regeln) beschrieben. Die 6 verpflichtenden abgeleiteten Regeln des Prozesses sind in Listing 19 abgebildet.

Listing 19: Abgeleitete Regeln der verhaltensorientieren Perspektive

```
%Regeln der verhaltensorientierten Perspektive
```

```
ensure sequence(Antrag stellen, Antrag intern genehmigen)
ensure sequence(Antrag stellen, Antrag pruefen (intern))
ensure sequence(Antrag pruefen (intern), Antrag pruefen (Verwaltung))
ensure sequence(Antrag pruefen (Verwaltung), Rueckmeldung bearbeiten)
ensure sequence(Rueckmeldung bearbeiten, Dienstreise archivieren)
ensure sequence(Flug buchen, Transfer buchen)
```

Die abgeleiteten Regeln zeigen beispielsweise, dass eine Antragsgenehmigung zuvor eine Antragstellung erfordert oder dass ein Antrag zuerst innerhalb der Forschungsgruppe überprüft wird, bevor er von zentralem Verwaltungspersonal geprüft wird. Zusätzlich zu den verpflichtenden Regeln werden lediglich tendenziell erfüllte, d.h. empfohlene Regeln abgeleitet. Diese Regeln zeigen beispielsweise, dass es zwar möglich ist Flug und Unterkunft vor einer Antragstellung zu buchen, es jedoch nicht empfohlen ist.

```
advice sequence(Antrag stellen, Flug buchen)
advice sequence(Antrag stellen, Unterkunft buchen)
advice sequence(Unterkunft buchen, Flug buchen)
```

Die Analyse mit der perspektivenübergreifenden Regelvorlage *roleSequence* liefert weitere Einblicke in die Ausführungsreihenfolge von Aktivitäten. Es wurden 6 *roleSequence*-Regeln abgeleitet. Die gewonnen Einblicke werden anhand der in Listing 20 abgebildeten Regeln exemplarisch ersichtlich.

Listing 20: Abgeleitete perspektivenübergreifende Regeln

```
%Perspektivenuebergreifende Regeln
advice sequence(Antrag stellen, Flug buchen)
ensure roleSequence(Antrag stellen, Flug buchen, Doktorand)
```

Abgeleitete perspektivenübergreifende Regeln

Dieser Teil des Modells betrachtet die Ausführungsreihenfolge der Aktivitäten unter Berücksichtigung der Eigenschaften ausführender Personen. Der obige Modellausschnitt zeigt, dass zwar eine Tendenz dazu besteht, dass Mitarbeiter zuerst einen Antrag stellen bevor Flugbuchungen durchgeführt werden, dies jedoch nicht verpflichtend ist (*empfohlene Aktivitäten-Sequenz*). Es existieren Fälle, in denen gewisse Mitarbeiter bereits einen Flug gebucht haben ohne zuvor einen Antrag zu stellen. Wird die Reihenfolge der Aktivitäten jedoch unter Berücksichtigung der ausführenden Personen und deren Rollen analysiert, so zeigt sich, dass Doktoranden stets einen Antrag gestellt haben bevor ein Flug gebucht wurde (*verpflichtende rollenbasierte Aktivitäten-Sequenz*). Während es Professoren frei steht einen Flug auch ohne genehmigten Antrag zu buchen, müssen sich Doktoranden an eine strikte Ausführungsreihenfolge halten.

Abgeleitete Abschlussbedingungen

Des Weiteren wurde analysiert, welche Aktivitäten in jeder Prozessinstanz verpflichtend abzuschließen waren. Dies wurde durch Analyse der Regelvorlage *complete(of A)* abgeleitet. Es zeigt sich, dass die fünf Aktivitäten, die in Listing 21 dargestellt sind, in jeder der aufgezeichneten Prozessinstanzen durchgeführt wurden. Diese sind daher in einem Meilenstein zusammengefasst.

Listing 21: Abgeleitete Prozess-Abschlussbedingungen

```
%Prozess-Abschlussbedingungen
milestone "Done": complete(of Antrag stellen) and
```

```
complete(of Antrag pruefen (intern)) and
complete(of Antrag pruefen (Verwaltung) and
complete(of Rueckmeldung bearbeiten) and
complete(of Dienstreise archivieren)
```

7.2.3 Genauigkeit und Vollständigkeit abgeleiteter Modelle

Evaluation der Genauigkeit und Vollständigkeit

Zur Evaluation der Genauigkeit und Vollständigkeit von abgeleiteten Modellen wurde ein Workshop mit prozessbeteiligten Personen durchgeführt. An der Diskussion nahmen insgesamt 7 Personen teil, wobei jede der beteiligten organisatorischen Gruppierungen, d.h. Doktoranden, Professoren, etc., durch mindestens eine Person vertreten war.

Evaluationsworkshop

Nachdem den Teilnehmern eine grobe Übersicht über die durch den DpilMiner abgeleiteten Modelle gegeben wird, wird jede der extrahierten Regeln separat betrachtet und einer der beiden folgenden Kategorien zugeordnet, d.h. i) *korrekt-positiv* (T_P: korrekt abgeleitete Regeln) oder ii) *inkorrekt-positiv* (F_P: fälschlicherweise abgeleitete Regeln). Eine Regel wird als inkorrekt bezeichnet, wenn sie entweder falsch oder irrelevant ist. Des Weiteren wurden in der Diskussionen, soweit möglich, fehlende Regeln identifiziert. Diese Regeln werden als *inkorrekt-negativ* (F_N: fälschlicherweise nicht abgeleitete Regeln) klassifiziert.

Klassifikation von Regeln

Zur Evaluation von Genauigkeit und Vollständigkeit des Verfahrens werden die Metriken **Genauigkeit** (*precision*) und **Trefferquote** (*engl. recall*) verwendet. Diese Metriken sind in Formel 5 definiert [109]. Diese beiden Gütemaße beeinflussen sich gegenseitig. Es ist nicht möglich, alle Metriken unabhängig voneinander zu betrachteten. Zur Veranschaulichung dieser Zusammenhänge ist es hilfreich, die Extremfälle zu betrachten:

Genauigkeit und Trefferquote

Werden sehr viele Regeln durch das Verfahren abgeleitet, ist die Wahrscheinlichkeit größer, dass sich auch alle relevanten Regeln darunter befinden. Es resultiert ein großer *Recall*-Wert. Allerdings wird dabei gleichzeitig auch die Inkorrekt-Positiv-Rate erhöht, da auch viele irrelevante Regeln unter den entdeckten Regeln sind. Das abgeleitete Modell hat somit eine geringe Genauigkeit. Werden hingegen wenige Regen abgeleitet, ist umgekehrt die Genauigkeit groß, allerdings auch die Trefferquote gering. Das **F-Maß** in Formel 5 kombiniert Genauigkeit und Trefferquote mittels des gewichteten harmonischen Mittels und stellt daher ein umfassenderes Güte-Kriterium für abgeleitete Modelle dar. Diese Metriken gelten als anerkannte Größen zum Vergleich von Verfahren im Information Retrieval. Die Werte Genauigkeit (*precision, P*), Trefferquote (*recall, R*) und *F* sind wie folgt definiert:

F-Maß

$$\text{Precision} = \frac{T_P}{T_P + F_P}, \qquad \text{Recall} = \frac{T_P}{T_P + F_N}, \qquad F = 2 \cdot \frac{P \cdot R}{P + R} \tag{5}$$

Als Grundlage der Evaluation dienen drei abgeleitete Modelle M1, M2 und M3, die auf unterschiedlichen Support-Werten in der Vorverarbeitungsphase und unterschiedlichen $\texttt{minConf}_S$-Werten während des Mining-Vorgangs ba-

sieren. Die Charakteristika der betrachteten Modelle sind in Tabelle 16 zusammenfassend dargestellt. M1 ist durch Anwendung des Verfahrens ohne Vorverarbeitungsphase und daher ohne jegliche Filterung abgeleitet worden. M2 stellt das im vorherigen Abschnitt beschriebene Modell dar. Zur Ableitung von M3 wurde der Schwellenwert für empfohlene Regeln von 85% auf 90% erhöht. Hierdurch wurden deutlich weniger Empfehlungen abgeleitet.

Tabelle 16: Charakteristika der drei abgeleiteten Modelle des Dienstreiseprozesses

	Modell 1	**Modell 2**	**Modell 3**
Konfiguration	keine Filterung	mäßige Filterung	starke Filterung
$\mathrm{minSupp}$ **(Vorverarbeitung)**	⊘	10%	10%
$\mathrm{minSupp}$ **(Regeln)**	⊘	20%	20%
$\mathrm{minConf}_H$	95%	95%	95%
$\mathrm{minConf}_S$	85%	85%	90%
Modell			
Anzahl Aktivitäten	10	9	9
Anzahl Identitäten	10	10	10
Anzahl Regeln	47	39	31

Die Klassifikation der Regeln der abgeleiteten Modelle in *korrekt abgeleitete Regeln* (T_P), *fälschlicherweise abgeleitete Regeln* (F_P) und *fehlende Regeln* (F_N) ist im Anhang A.1.3 dieser Arbeit gegeben. Die Ergebnisse der Evaluation, sowie die resultierenden Güte-Metriken sind zusammenfassend in Tabelle 17 dargestellt.

Tabelle 17: Metriken der drei abgeleiteten Modelle des Dienstreiseprozesses

Wert/Metrik	**Modell 1**	**Modell 2**	**Modell 3**
Konfiguration	keine Filterung	mäßige Filterung	starke Filterung
T_P (*korrekt-positiv*)	40	34	28
F_P (*inkorrekt-positiv*)	7	5	3
F_N (*inkorrekt-negativ*)	0	6	12
Genauigkeit (*precision*)	0,85	0,87	0,9
Trefferquote (*recall*)	1,0	0,85	0,7
F-Maß	**0,92**	**0,86**	**0,78**

Betrachten wir exemplarisch die Evaluationsergebnisse für M2. Insgesamt wurden 34 der 39 extrahierten Regeln als relevant eingestuft (T_P), wohingehen 5 Regeln als inkorrekt (F_P) klassifiziert wurden. Dies entspricht einem *Precision*-Wert von 0,87. Betrachten wir die Menge der abgeleiteten Regeln, die von prozessbeteiligten Personen als inkorrekt eingestuft wurden. Diese sind in Listing 22 dargestellt.

Listing 22: Menge der inkorrekt abgeleiteten Regeln in M2

```
%Menge der inkorrekt abgeleiteten Regeln (FP)
ensure sequence(Flug buchen,Transfer buchen)
advice sequence(Unterkunft buchen,Transfer buchen)
advice sequence(Unterkunft buchen,Dienstreise archivieren)
```

```
ensure binding(Antrag pruefen(intern),Rueckmeldung bearbeiten)
ensure binding(Antrag pruefen(intern),Dienstreise archivieren)
```

In allen aufgezeichneten Prozessinstanzen, in denen ein Transfer gebucht wird, wird demnach zuvor ein Flug gebucht. Die Regel spiegelt zwar das Verhalten der aufgezeichneten Instanzen wieder, wurde von den prozessbeteiligten Personen jedoch als inkorrekt eingestuft. Auch die beiden Empfehlungen eine Unterkunft vor einem Transfer zu buchen und bevor die Unterlagen archiviert werden, wurden im Workshop als inkorrekt eingestuft. Die beiden *binding*-Regeln wurden ebenso als falsch eingestuft, da der Antrag nicht zwingend von der Person geprüft werden muss, die auch die Rückmeldung von der Verwaltung bearbeitet und die Dienstreise archiviert. Es ist hervorzuheben, dass diese Regeln lediglich das im betrachteten Ereignisprotokoll aufgezeichnete Verhalten widerspiegeln. Die fünf abgeleiteten irrelevanten Regeln zeigen, dass die häufigkeitsbasierte Betrachtung zwar größtenteils relevante Zusammenhänge liefert, jedoch auch häufig auftretende aber irrelevante Regeln extrahiert werden.

False-negative Regeln in M2

Des Weiteren wurden 6 fehlende Regeln (F_N) in der Diskussion identifiziert. Dies ist darauf zurückzuführen, dass die Aktivität "Zug buchen" durch die Filterung in der Vorverarbeitungsphase entfernt wurde, da diese in weniger als 10% aller aufgezeichneten Spuren auftritt. M2 enthält also weder die Definition der Aktivität noch entsprechende Kontrollfluss- oder Zuweisungsregeln. Die fehlenden Regeln sind in Listing 23 gegeben.

Listing 23: Menge der fehlenden Regeln in M2

```
%Fehlende Regeln (FN)
advice sequence(Antrag stellen,Zug buchen)
advice sequence(Antrag pruefen(intern),Zug buchen)
ensure binding(Antrag stellen,Zug buchen)
ensure orgDistM(Antrag intern genehmigen,Zug buchen,supervisor)
ensure roleSequence(Antrag intern genehmigen,Zug buchen,Doktorand)
ensure roleSequence(Antrag pruefen(intern),Zug buchen,Doktorand)
```

Vergleich der Güte abgeleiteter Modelle

Insgesamt ergibt sich für M2 daher ein *Recall*-Wert von 0,85 und ein kombinierter Gütewert F von 0,86. Der Vergleich der drei abgeleiteten Modelle in Tabelle 17 zeigt, dass M1 den höchsten F-Wert aufweist. Dadurch, dass keine seltenen Zusammenhänge gefiltert wurden, sind alle relevanten Zusammenhänge im Modell enthalten (*Recall=1,0*). Hier wurden jedoch auch 7 irrelevante Regeln abgeleitet, weshalb M1 einen Genauigkeitswert von 0,85 aufweist. Zur Ableitung von M3 wurde, neben einer Vorverarbeitungsphase, vor allem die Extraktion von Empfehlungen durch eine Erhöhung des $minConf_S$-Wertes reduziert. Diese wurden von den Workshop-Teilnehmern jedoch als relevant und wichtig eingestuft, weshalb in M3 12 Regeln fehlen. Auch wenn M3 die höchste Genauigkeit aufweist, ergibt sich aufgrund der relativ niedrigen Trefferquote daher der niedrigste F-Wert unter den drei Modellen von 0,78.

7.2.4 Laufzeit, Vorverarbeitung und Modell-Nachbearbeitung

In diesem Abschnitt wird die Laufzeit des Analysevorgangs und deren Verbesserung durch die Vorverarbeitungsphase evaluiert. Des Weiteren betrachten wir die Menge an extrahierten Regeln vor und nach der Modell-Nachbearbeitung. Hierzu analysieren wir für beide Konfigurationen Θ_1 und Θ_2 (i) die Anzahl der zu überprüfenden Regelkandidaten ausgehend von verschiedenen *minSupp*-Werten während der Vorverarbeitungsphase, (ii) die Anzahl der abgeleiteten Regeln vor der Modell-Nachbearbeitung, (iii) die Anzahl der resultierenden Regeln nach der Modell-Nachbearbeitung, (iv) die benötigte Zeit zum Aufbau des Rete-Netzwerks und (v) die benötigte Zeit für den eigentlichen Analysevorgang. Als Referenz-Schwellenwerte wählen wir wiederum $\mathrm{minSupp} = 20\%$, $\mathrm{minConf_S} = 85\%$ und $\mathrm{minConf_H} = 95\%$. Die Laufzeittests werden anhand der seriellen Regelüberprüfung durchgeführt. Auf diese Weise ist das Verfahren besser mit existierenden Techniken vergleichbar. Die resultierenden Werte der Analyse mit der Menge Θ_1 an organisatorischen Regelvorlagen sind in Abbildung 40 dargestellt.

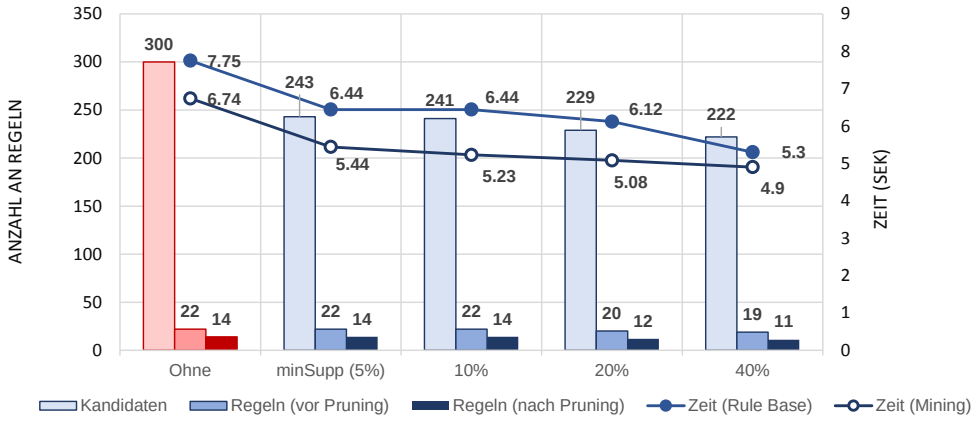


Abb. 40: Analyse des Dienstreiseprozesses mit organisatorischen Regelvorlagen

Die Werte der Analyse mit der Menge an verhaltensorientierten bzw. perspektivenübergreifenden Regelvorlagen Θ_2 sind in Abbildung 41 visualisiert.

Die Auswertung der beiden Diagramme liefert die im folgenden Abschnitt beschriebenen Erkenntnisse:

Überschaubare Anzahl an Regeln

- Trotz der großen Anzahl an Regelkandidaten wird bei beiden Konfigurationen eine überschaubare Anzahl an Regeln entdeckt. Abbildung 40 zeigt, dass bei Konfiguration Θ_1 und $\mathrm{minSupp} = 5\%$ beispielsweise 243 Regelkandidaten lediglich 14 tatsächlich abgeleitete Regeln gegenüberstehen. Die überschaubare Anzahl an resultierenden Regeln belegt die generelle Anwendbarkeit des Verfahrens und die Überschaubarkeit von resultierenden Prozessmodellen.

Nutzen der Vorverarbeitungsphase

- In Diagramm 41 wird der Vorteil der Vorverarbeitungsphase hinsichtlich der Laufzeit des Verfahrens besonders ersichtlich. Ohne Vorverar-

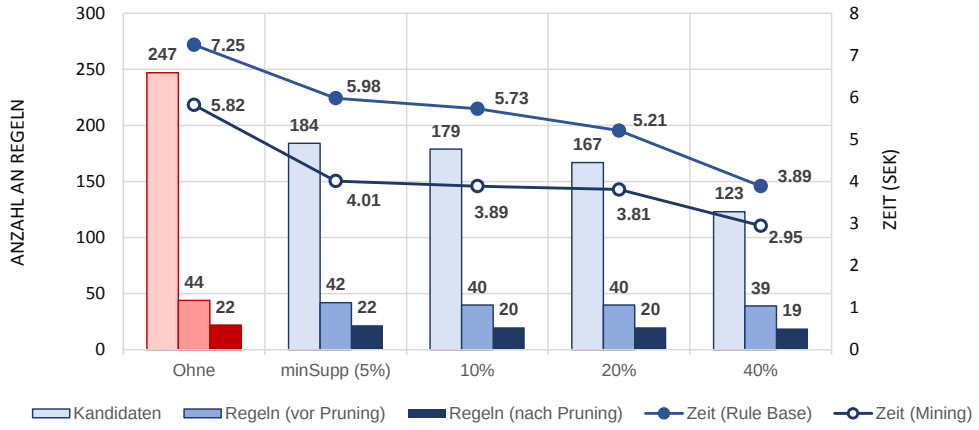


Abb. 41: Analyse des Dienstreiseprozesses mit verhaltensorientierten Regelvorlagen

beitungsphase sind 247 Regelkandidaten zu überprüfen. Mit steigendem *minSupp*-Wert nimmt die Anzahl der zu überprüfenden Regelkandidaten deutlich ab (184 Kandidaten bei $\mathtt{minSupp} = 5\%$ zu 123 Kandidaten bei $\mathtt{minSupp} = 40\%$). Die benötigte Zeit zum (einmaligen) Aufbau des Rete-Netzwerks sinkt dadurch von 7,25 auf 3,89 Sekunden und die Laufzeit des Analyse-Vorgangs von 5,82 auf 2,95 Sekunden. Die Anzahl an tatsächlich extrahierten Regeln bleibt jedoch nahezu gleich. Es werden lediglich 3 Regeln weniger abgeleitet als ohne Vorverarbeitung.

Entfernen von Redundanzen

- Bei beiden Analysen kann die Anzahl an extrahierten Regeln durch die Nachbearbeitung des Modells, d.h. durch das Entfernen von redundanten Regeln, deutlich reduziert werden. Auf diese Weise werden bei der Analyse mit Θ_1 jeweils 8 redundante Regeln entfernt. Bei einem $\mathtt{minSupp}$ von 40% entspricht dies beispielsweise einer Reduzierung der Regelmenge von 43%. Da die betrachteten organisatorischen Zusammenhänge nicht transitiv reduziert werden können, handelt es sich hierbei um eine Reduzierung der Regelmenge auf Basis von Regelhierarchien wie in Abschnitt 6.2 beschrieben. Bei der Analyse mit Θ_2 kann die Regelmenge durch die Nachbearbeitung nahezu halbiert werden. Die abgeleitete Menge an *sequence*-Regeln und *roleSequence*-Regeln lässt sich sowohl transitiv (vgl. Abschnitt 6.3) als auch auf Basis von Regelhierarchien reduzieren.

Adäquate Laufzeit

- Das vollständige Ereignisprotokoll mit 128 Instanzen und 2104 Ereignissen wird bei allen Konfigurationen in wenigen Sekunden analysiert. Die absoluten Laufzeitangaben belegen zwar die generelle Anwendbarkeit des Verfahrens, sind jedoch erst durch einen relativen Vergleich mit den Laufzeiten vergleichbarer Verfahren aussagekräftig. Ein Vergleich mit den Laufzeiten des etablierten *DeclareMiners* erfolgt deshalb in Kap. 8.2.1. Zusätzlich ist noch hervorzuheben, dass die Berechnungen während der Vorverarbeitungsphase, d.h. die Anwendung des Apriori-Algorithmus, bei allen Konfigurationen stets deutlich weniger als eine Sekunde benötigten.

Die Laufzeit der Vorverarbeitungsphase ist demnach vernachlässigbar gering.

7.3 ANALYSE EINES KLINISCHEN PROZESSES

Neben der Analyse des Dienstreiseprozesses wird der vorgestellte Ansatz auch anhand eines Ereignisprotokolls aus dem medizischen Bereich evaluiert. Bei dem im folgenden Abschnitt beschriebenen Protokoll handelt es sich um die Aufzeichnungen eines Behandlungsprozesses von Krebspatienten in einem niederländischen Klinikum. Dieses Protokoll[5] ist im Rahmen der *Business Process Intelligence Challenge* (BPI Challenge 2011) zur Verfügung gestellt worden und dient zur Evaluation und zum Vergleich von Process Mining-Algorithmen.

7.3.1 Beschreibung und Filterung des Ereignisprotokolls

Das als XES-Datei zur Verfügung gestellte Ereignisprotokoll beinhaltet insgesamt 1143 Spuren und 150291 Ereignisse, die 624 verschiedenen Aktivitäten zuzuordnen sind. Das Log enthält Aktivitäten der Behandlung von Krebspatienten verschiedener Art und in unterschiedlichen Stadium. Jede Spur des Logs ist dabei einem Patienten zuzuordnen. Neben der zugehörigen Aktivität und weiteren Datenattributen enthält jedes Ereignis ein *org:group*-Attribut, welches angibt, welche Abteilung die entsprechende Aktivität durchgeführt hat. Das Log enthält lediglich *Complete*-Ereignisse der Aktivitäten. Ein exemplarisches Ereignis des XES-Logs ist in Listing 24 visualisiert. Der Ausschnitt beschreibt das *Complete*-Ereignis der Aktivität "haemoglobin photoelectric - urgent" (dringende photoelektrische Hämogloblin Analyse). Die Aktivität ist dabei in der Abteilung "General Lab Clinical Chemistry" durchgeführt worden.

Listing 24: Exemplarisches Ereignis im klinischen Log

```
<event>
    <string key="org:group" value="General Lab Clinical Chemistry"/>
    <string key="concept:name" value="haemoglobin photoelectric - urgent"/>
    <date key="time:timestamp" value="2005-03-21T00:00:00.000+01:00"/>
    <string key="lifecycle:transition" value="complete"/>
    ...
</event>
```

Aufgrund der großen Anzahl an Aktivitäten und der enormen Heterogenität der Behandlungsfälle ist eine Analyse des vollständigen Logs ohne Vorverarbeitung nahezu unmöglich. Zur Bereitstellung eines analysierbaren Logs wird das ursprüngliche Protokoll vorverarbeitet. Wir verwenden hierzu die Methode aus [28]. Zu jeder Prozessinstanz sind zu Beginn verschiedene Datenattribute gegeben auf Basis derer das Log in kleinere, homogenere Ereignisprotokolle aufgeteilt werden kann. Jede Spur (*trace*) enthält beispielsweise ein Attribut *Dia-*

5 doi:10.4121/uuid:d9769f3d-0ab-4fb8-803b-0d1120ffcf54

gnosis code oder *Diagnosis*, welches die Krebsdiagnose des Patienten beschreibt (Listing 25).

Listing 25: Diagnose Code-Attribut im klinischen Log

```
<trace>
    <string key="Diagnosis code" value="M11"/>
    ...
    <event>
        ...
</trace>
```

Das Log beinhaltet Behandlungsprozesse zu 11 verschiedenen Krebsdiagnosearten. Der erste Schritt zur Erzeugung eines homogeneren Logs ist die Aufteilung des Logs nach Diagnosearten, d.h. beispielsweise anhand des Attributs *Diagnosis code*. Des Weiteren werden im ursprünglichen Log *dringende* (*engl. urgent*) und *nicht-dringende* Fälle unterschieden. Das Log enthält beispielsweise die Aktivität "haemoglobin photoelectric" als auch die Aktivität "haemoglobin photoelectric - urgent". Neben der Diagnoseart kann das Log zusätzlich in dringende und nicht-dringende Fälle aufgeteilt werden. Als dringende Fälle werden dabei die Spuren bezeichnet, in denen mindestens eine Aktivität den Zusatz "urgent" enthält. Neben den beschriebenen Eigenschaften einzelner Behandlungsfälle existieren noch weitere Attribute, auf Basis derer homogene Sub-Protokolle erzeugt werden können. Wie in [28] werden in dieser Arbeit emeplarisch die folgenden Vorverarbeitungsschritte durchgeführt:

- Auswahl aller Prozessinstanzen, die den Diagnose-Code "M11" aufweisen. Insgesamt existieren 162 Behandlungsfälle dieser Krebsart im ursprünglichen Log. Das gefilterte Log enthält schließlich noch 11280 Ereignisse und 207 verschiedene Aktivitäten. Wir bezeichnen das resultierende Protokoll als *Log 1*.

- Erzeugung von separaten Logs für dringende und nicht-dringende Fälle auf Basis des nach M11-gefilterten Logs. Von 162 Fällen sind 137 nicht-dringend mit 6225 Ereignissen sowie 143 Aktivitäten und 25 Fälle dringend mit 5055 Ereignissen sowie 173 Aktivitäten. Wir bezeichnen das resultierende Protokoll der dringenden Fälle als *Log 2*.

- Fokussierung auf Ereignisse, die in Abteilung "General Lab Clinical Chemistry" durchgeführt worden sind. Durch diese Filterung entsteht ein Log mit 2739 Ereignissen und 93 verschiedenen Aktivitäten. Wir bezeichnen das resultierende Protokoll als *Log 3*.

Log 3 enthält daher lediglich dringende Fälle bei Krebsdiagnose "M11" die in der Abteilung "General Lab Clinical Chemistry" durchgeführt worden sind. In den folgenden Abschnitten wird ersichtlich, dass bereits die Analyse dieses kleinen Log-Ausschnitts zu sehr komplexen Modellen führt.

7.3.2 Analyse durch klassische Verfahren

Betrachten wir zuerst die Analyse von *Log 3* mit einem klassischen Process Mining-Verfahren wie dem α-Algorithmus der ProM-Implementierung (Version 6.3)[6]. Das resultierende Petrinetz ist in Abbildung 42 dargestellt und im Anhang A.2 vergrößert. Das Modell enthält Knoten für die 93 Aktivitäten des Behandlungsprozesses und unüberschaubar viele Sequenzflusslinien da zahlreiche verschiedene Ablaufmöglichkeiten existieren, welche alle explizit im abgeleiteten Petrinetz abgebildet werden. Eine adäquate Interpretation des Modells erscheint nahezu unmöglich.

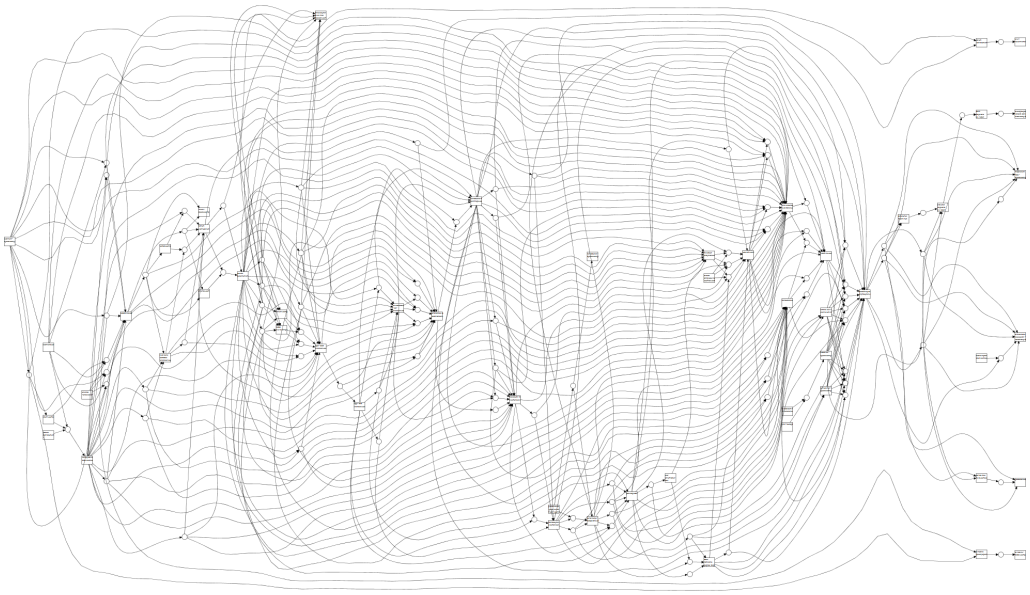
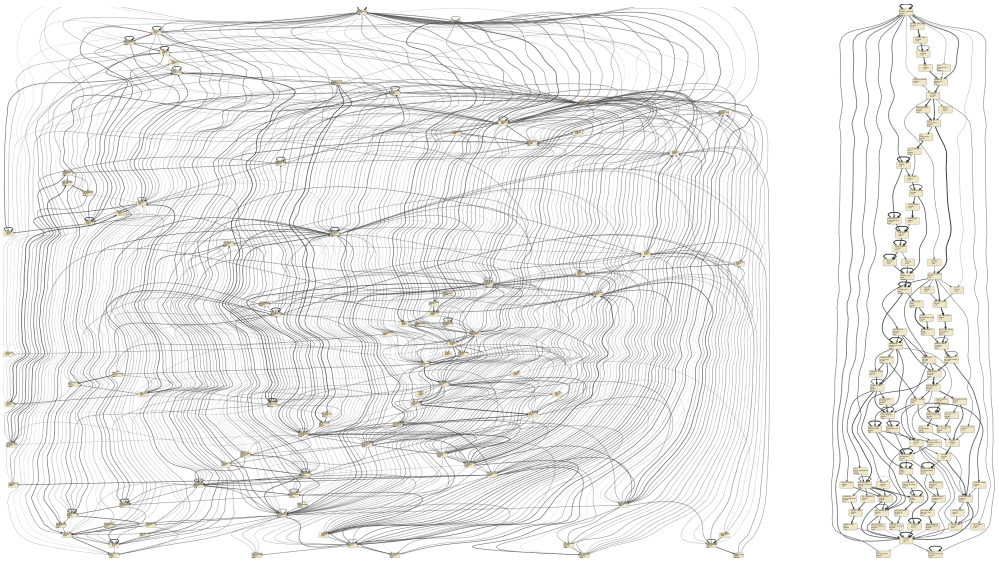


Abb. 42: Analyse eines klinischen Prozesses mit dem α-Algorithmus

Des Weiteren wird das gegebene Log auch mit der ProM-Implementierung (Version 6.3) des Fuzzy Miners analysiert. Durch die Anwendung des Fuzzy Miners mit zwei verschiedenen Schwellenwerten wird untersucht, wie sich die Abstraktion von seltenen Kanten auf die Lesbarkeit und die Ausdrucksstärke des Modells auswirkt. Die resultierenden Fuzzy-Modelle sind in Abbildung 43 dargestellt. Abbildung 43a zeigt das abgeleitete Fuzzy-Modell bei nahezu keiner Kantenfilterung (*edge cut-off = 0,9*). Auf Basis dieses Modells können rund 96% der im Log enthaltenen Instanzen abgbebildet werden. Das Modell ist jedoch ähnlich unübersichtlich wie das abgeleitete Petrinetz durch Anwendung des α-Algorithmus, weil nahezu sämtliche Ablaufpfade explizit enthalten sind. Durch eine Filterung der Kanten (*edge cut-off = 0,265*) kann die Übersichtlichtkeit des abgeleiteten Modells deutlich erhöht werden (Abbildung 43b). Nach der Filterung spiegelt das Modell jedoch lediglich noch rund 73% der im Log enthaltenen Ablaufpfade wieder. Das Modell kann für einen generellen Über-

6 Verfügbar unter www.promtools.org

blick über den Prozess gut geeignet sein. Vor allem im Bereich klinischer Prozesse können auf diese Weise jedoch wichtige Zusammenhänge verloren gehen.



(a) Keine Kantenfilterung

(b) Mit Kantenfilterung

Abb. 43: Analyse des klinischen Prozesses mit dem Fuzzy Miner

7.3.3 Analyse mit dem DpilMiner

Betrachten wir nun die Analyse durch Anwendung des DpilMiners. Zuerst wird *Log 2* hinsichtlich der organisatorischen Perspektive analysiert. Ziel ist es abzuleiten, welche Aktivitäten in dringenden M11-Behandlungsprozessen von welcher Abteilung durchgeführt worden sind. Da das Log nur ein *org:group*-Attribut und *Complete*-Ereignisse enthält, ist zur Analyse weder die ursprünglich definierte *direct*-Vorlage noch die *role*-Vorlage verwendbar. Bei der Analyse werden Abteilungen wie konkrete Ressourcen behandelt. Zur Analyse wird eine neue Regelvorlage *group* definiert:

```
group(A, G) iff
complete(of A1) implies complete(of A by G)
```

Log 2 wird in der Vorverarbeitungsphase mit einem $\mathtt{minSupp}$-Wert von 5% gefiltert, so dass nur Aktivitäten einbezogen werden, die in mindestens 5% der aufgezeichneten Instanzen auftreten. Als Schwellenwerte zur Regelklassifikation werden wie wie zuvor $\mathtt{minSupp}$=20%, $\mathtt{minConf}_H$=95% und $\mathtt{minConf}_S$=85% gesetzt. Die Analyse der *group*-Vorlage liefert insgesamt 74 verpflichtende gruppenbasierte Zuweisungsregeln und 1 empfohlene Regel. Von 136 Aktivitäten werden demnach 74 stets von der gleichen Abteilung durchgeführt. Die meisten Aktivitäten sind dabei der Abteilung "General Lab Clinical Chemistry" zugeordnet (47 Regeln). Zum Aufbau des Rete-Netzwerks werden dabei 4,39

Sekunden benötigt. Der eigentliche Mining-Vorgang wird in 7,9 Sekunden abgeschlossen. Die konkreten Zuweisungsregeln sind im Anhang A.2.2 dieser Arbeit visualisiert. In Listing 26 sind ausgewählte Regeln abgebildet. Im Modell wird ersichtlich, dass vor allem die Bluttests stets im chemischen Labor stattfinden. Die Erhebung der Geburtshilfe-Kartengebühr wird vorzugsweise in der Geburtshilfe-Frauenklinik (*engl. Obstetrics Gynaecology Clinic*) durchgeführt, in wenigen Fällen jedoch auch in der Pflegestation (*engl. Nursing Ward*)

Listing 26: Abgeleitete Zuweisungsregeln des Klinikprozesses

```
%Regeln der organisatorischen Perspektive
ensure group(Glucose, General Lab Clinical Chemistry)
ensure group(Ureum, General Lab Clinical Chemistry)
ensure group(Sediment, General Lab Clinical Chemistry)
ensure group(Bicarbonaat, General Lab Clinical Chemistry)
ensure group(Calcium, General Lab Clinical Chemistry)
ensure group(Creatinine, General Lab Clinical Chemistry)
...
advice group(Obstetrics Card Fee, Obstetrics Gynaecology Clinic)
```

Betrachten wir nun die Analyse von *Log 3* durch den DpilMiner. Wie im vorherigen Abschnitt gezeigt, liefern klassische, prozedurale Miner nahezu unlesbare Modelle. Nur durch eine starke Filterung von selten auftretenden Pfaden kann das extrahierte Modell in eine überschaubarere Form transformiert werden. Wird die verhaltensorientierte Perspektive anhand der *sequence*-Vorlage mit dem DpilMiner analysiert, so werden insgesamt 80 verpflichtende Regeln abgeleitet. Es erfolgt dabei keine Vorfilterung. Anders als beim FuzzyMiner können alle aufgezeichneten Prozessinstanzen anhand dieser Regeln lückenlos beschrieben werden. In Anbetracht der zu beschreibenden zeitlichen Abhängigkeiten von 93 Aktivitäten und der unzähligen Sequenzflusslinien in klassischen Modellen ist die Anzahl an extrahierten Regeln durchaus überschaubar. Sämtliche abgeleitete zeitliche Abhängigkeit sind im Anhang A.2.2 abgebildet. Betrachten wir exemplarisch die zeitlichen Abhängigkeiten der Aktivität "Annahme Labor" (*niederl. "aanname laboratoriumonderzoek"*). Diese können durch lediglich 9 *sequence*-Regeln beschrieben werden und sind in Listing 27 dargestellt.

Listing 27: Abhängigkeiten von "Annahme Labor" im Klinikprozess

```
ensure sequence(Annahme Labor, Sediment - dringend)
ensure sequence(Annahme Labor, Bilirubine (konjugiert))
ensure sequence(Annahme Labor, Glucose)
ensure sequence(Annahme Labor, Haemoglobin photoelektrisch)
ensure sequence(Annahme Labor, Creatin)
ensure sequence(Annahme Labor, Blutgruppe Rhesus-Faktor)
ensure sequence(Annahme Labor, Natrium photometrisch (dringend))
ensure sequence(Annahme Labor, Kalium photometrisch (dringend))
ensure sequence(Annahme Labor, Creatin (dringend))
```

Betrachtet man im Vergleich dazu die gleiche Aktivität im Fuzzy Modell in Abbildung 43a ohne Kantenfilterung. In Abbildung 44 ist diese Aktivität und

deren eingehende und ausgehende Sequenzflusslinien vergrößert dargestellt. Soweit identifizierbar enthält das Modell zur Abbildung aller möglichen Pfade mindestens 25 eingehende bzw. ausgehende Kanten.

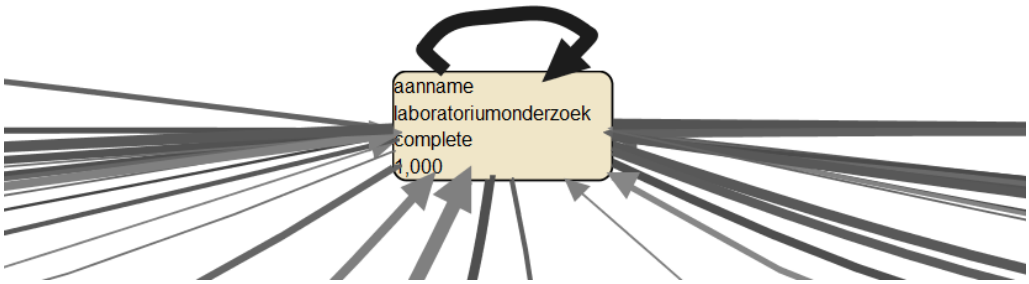


Abb. 44: Exemplarischer Ausschnitt des Fuzzy Modells (Log 3)

Wir betrachten wiederum Laufzeit und die Anzahl an Regelkandidaten und abgeleiteten Regeln für unterschiedliche Konfigurationen. Die resultierenden Werte sind in Abbildung 45 visualisiert. Die Laufzeiten können durch die Vorverarbeitungsphase ohne Verlust von Regeln wiederum drastisch reduziert werden. Während die Analyse von *Log 3* ohne Vorverarbeitung durch den Apriori-Algorithmus 258 Sekunden benötigt, erfolgt der Mining-Vorgang bei einem minSupp von 20% in 77,94 Sekunden und daher in nahezu einem Drittel der Zeit. Die Laufzeiten belegen außerdem wiederum die generelle Anwendbarkeit des Verfahrens. Erst bei einem minSupp-Wert von 40% nimmt die Zahl an abgeleiteten Regeln ab.

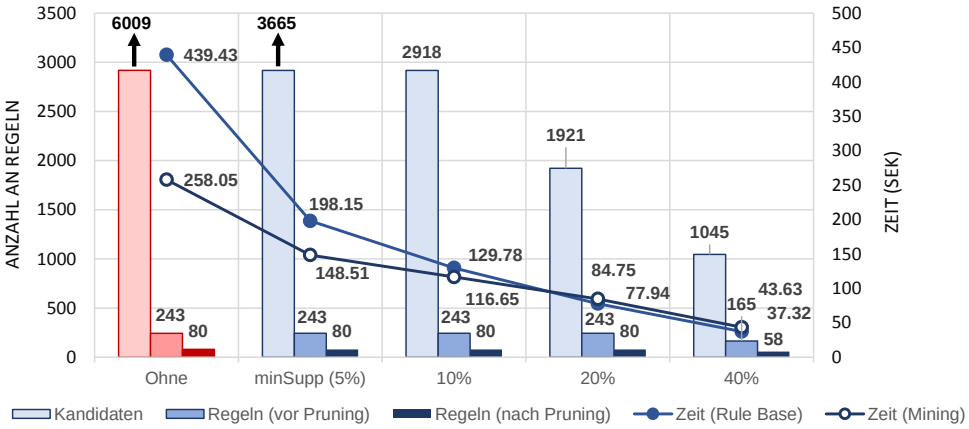


Abb. 45: Laufzeiten bei Analyse des klinischen Prozesses

7.4 ZUSAMMENFASSUNG

In diesem Kapitel wurde die **prototypische Implementierung** des vorliegenden Ansatzes in Form des **DpilMiners** vorgestellt. Zur **praktischen Evaluation** des Ansatzes wird die Anwendung auf Ereignisprotokolle eines universitären

Dienstreiseabwicklungsprozesses und eines klinischen Behandlungsprozesses beschrieben. Während klassische Verfahren wie der α-Algorithmus oder der Fuzzy Miner aus den Ereignisprotokollen unübersichtliche und schwer verständliche Modelle extrahiert, können die verhaltensorientierten Zusammenhänge durch abgeleitete, regelbasierte DPIL-Modelle des vorliegenden Ansatzes mit nur wenigen Regeln beschrieben werden. Zusätzlich kann durch die Analyse des DpilMiners ein Einblick in die komplexen organisatorischen Zusammenhänge der agilen Prozesses erlangt werden. Laufzeitmessungen belegen die praktische Anwendbarkeit des Verfahrens.

Neben der wissenschaftlichen Evaluation wurde der DpilMiner auch von Seiten der Wirtschaft und Industrie im Rahmen des Projektes "Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement" (*KpPQ*) praxisnah evaluiert. In mehreren Workshops wurde, in Kooperation mit Industriepartnern, die Funktionalität des Analysewerkzeugs getestet und dessen Anwendbarkeit in der Praxis erprobt.

8 VERWANDTE ARBEITEN

INHALT

Diese Dissertation befasst sich mit der Entwicklung eines integrierten Process Mining-Verfahrens für agile, personenbezogene Prozesse. Die Relevanz dieser Klasse von Prozessen wird durch zahlreiche wissenschaftliche Arbeiten und Problemfälle aus Wirtschaft und Industrie motiviert. Zur Abgrenzung von existierenden Ansätzen, welche agile Prozesse bzw. den organisatorischen Aspekt von Prozessen betreffen, müssen zwei Forschungsbereiche betrachtet werden:

- **Regelbasierte bzw. deklarative Process Mining-Verfahren**, die anstatt prozeduraler, flussorientierter Modelle, regelbasierte bzw. deklarative Modelle erzeugen bzw. die Konformität von Logs und gegebener Modelle überprüfen.
- **Organisatorische Process Mining-Verfahren**, die Ereignisprotokolle aus einer ressourcen- bzw. organisatorischen Perspektive analysieren.

Während bislang lediglich jeder Forschungsbereich separat betrachtet wurde, wird in dieser Dissertation ein Analyseverfahren vorgestellt, welches beide Strömungen vereint und regelbasierte Prozessmodelle unter Einbeziehung von Ressourcen und organisatorischen Zusammenhängen erzeugt. Nachdem noch einmal kurz eine Abgrenzung von klassischen Process Mining-Verfahren erfolgt, werden nachfolgend verwandte Arbeiten aus beiden Teilbereichen vorgestellt und deren Defizite aufgezeigt.

8.1 KLASSISCHE PROCESS MINING-VERFAHREN

Während des letzten Jahrzehnts wurden eine Reihe von Algorithmen zur automatisierten Ableitung von Prozessmodellen aus prozessorientierten Ereig-

nisprotokollen entwickelt [46]. Betrachten wir zuerst Methoden zur Modell-Entdeckung (*Process Discovery*). Diese Process Mining-Techniken sind in der Lage Prozessmodelle ohne Vorwissen ausgehend von einem Ereignisprotokoll zu entdecken. Die wichtigsten Vertreter prozeduraler Process Discovery-Verfahren sind der in Abschnitt 2.4.1 beschriebene α-Algorithmus [7] der der HeuristicMiner [137], der genetische Process Mining-Algorithmus [86], das Verfahren nach Bergenthum [26] oder der FuzzyMiner [57], [58]. Alle diese traditionellen Methoden basieren auf prozeduralen Prozessmodellierungssprachen, in denen alle im Ereignisprotokolle auftretenden Ablaufpfade explizit im resultierenden Modell dargestellt werden. Wendet man diese Verfahren auf die Ereignisprotokolle von agilen, besonders variantenreichen Prozessen an, werden typischerweise schwer verständliche und teilweise unlesbare Modelle abgeleitet. Ein Beispiel ist das "Spaghetti-Modell" in Abbildung 15. Beim FuzzyMiner wird die verbesserte Lesbarkeit des Modells letztendlich nur durch "Löschen" von (potentiell wichtigen) Informationen erreicht.

Klassische Process Mining-Verfahren

8.2 REGELBASIERTE PROCESS MINING-VERFAHREN

Der Flexibilität bzw. Agilität von realen Prozessen, welche sich auch in der Variabilität von Ereignisprotokollen zeigt, kann auch auf eine völlig andere Art und Weise begegnet werden. Anstatt die Komplexität extrahierter Modelle durch Abstraktion von seltenen Pfaden zu verringern, werden in **regelbasierten Process Mining-Verfahren** als Zielsprache regelbasierte Sprachen verwendet. Hierdurch können besonders variantenreiche Prozesse durch eine (überschaubare) Menge an Regeln beschrieben werden.

In den letzten Jahren wurden einige regelbasierte Process Mining-Verfahren entwickelt, welche ausführbare Prozessmodelle extrahieren und als Zielsprache *Declare* (in Abschnitt 3.2.1 detailliert beschrieben) verwenden. Im Wesentlichen fokussieren alle Ansätze lediglich die verhaltensorientierte Perspektive mit einigen Erweiterungen für die datenorientierte Perspektive. Die organistorische Perspektive und deren Einfluss auf den Prozessablauf wird hingegen nicht betrachtet.

8.2.1 DeclareMiner und Erweiterungen

Erster Ansatz des DeclareMiners

Der erste Ansatz des *DeclareMiners* [81] verwendet eine Menge von vordefinierten Declare-Regelvorlagen, welche mit allen möglichen, im betrachteten Ereignisprotokoll auftretenden Aktivitäten instanziiert werden. Da Declare auf linearer temporaler Logik (LTL) basiert, sind diese Regelvorlagen in LTL definiert. Ähnlich wie im Ansatz der vorliegenden Arbeit wird anschließend für jede Regel der entstehenden Menge an Regelkandidaten, deren Gültigkeit für jede aufgezeichnete Prozessinstanz überprüft. Der DeclareMiner verwendet für die Überprüfung der LTL-Regeln den *LTL-Checker* [1].

LTL-Checker

Der Parameter *Prozent an Prozessinstanzen (engl. Percentage of Instances, PoI)* kann dazu verwendet werden, auch Regeln abzuleiten, die nicht in allen Prozessinstanzen erfüllt sind. Mit PoI = 95% werden beispielsweise auch Regeln ins Modell übernommen, die in lediglich 95% der analysierten Instanzen erfüllt sind. Der Parameter wird verwendet um das resultierende Modell durch Ausnahmen und Fehler nicht zu verfälschen. Auch in diesem Ansatz wird zwischen trivial und nicht-trivial erfüllten Prozessinstanzen unterschieden. Der Parameter *Prozent an interessanten Nachweisen (engl. (Percentage of Interesting Witnesses, PoIW)* wird dazu verwendet, nur Regeln abzuleiten, die in genügend vielen Instanzen auftreten. Wird PoIW = 10% gesetzt, werden beispielsweise nur Regeln entdeckt, die in mind. 10% der Instanzen des Ereignisprotokolls nicht trivial erfüllt oder verletzt sind.

Schwächen des DeclareMiners

Mit dem DeclareMiner ist es bislang nicht möglich Regeln unterschiedlicher Modalitäten abzuleiten. Eine abgeleitete Regel im resultierenden Modell ist daher **immer verpflichtend**. Ist eine Regel zwar nicht immer, jedoch tendenziell erfüllt, geht diese Information verloren und ist nicht im Modell enthalten. Es ist wichtig zu erwähnen, dass Declare grundsätzlich die Möglichkeit bietet, mittels optionaler Regeln, Empfehlungen abzubilden. Der DeclareMiner betrachtet **lediglich die verhaltenorientierte Perspektive** während der Protokoll-Analyse. Sämtliche Suchvorlagen des DeclareMiners betreffen zeitliche Abhängigkeiten zwischen Aktivitäten und lassen organisatorische Zusammenhänge unbeachtet. Dies ist durch die **Darstellungsausrichtung des DeclareMiners** auf Declare-Modelle begründet. Wie in Abschnitt 3.2.1 erläutert, können in Declare **keine organisatorischen Zusammenhänge** abgebildet werden. Es ist daher weder möglich organisatorische Zuweisungsregeln abzuleiten, noch den Einfluss der organisatorischen Perspektive auf den Kontrollfluss, d.h. perspektivenübergreifende Zusammenhänge.

Implementierung des DeclareMiners in ProM

Eine Implementierung des DeclareMiners ist in dem Process Mining-Tool *ProM* enthalten [77]. Um die Laufzeit des vorliegenden Verfahrens bzw. der Implementierung im DpilMiner mit dem DeclareMiner zu vergleichen, wenden wir diesen auf das Ereignisprotokoll aus Abschnitt 7.2 an. Das Log wird lediglich mit der *precedence*-Regelvorlage von Declare analysiert, welche der *sequence*-Vorlage in DPIL entspricht. Betrachtet man die Laufzeit der Analyse, so zeigt sich, dass der DpilMiner das Ereignisprotokoll deutlich schneller analysiert als der DeclareMiner obwohl 4 bzw. 2 Regelvorlagen analysiert werden. Mit Standardeinstellungen benötigt der DeclareMiner zur vollständigen Analyse des Logs 14,85 Sek. Der DpilMiner analysiert das Protokoll trotz der komplexeren Konfigurationen wesentlich schneller. Die Laufzeitprobleme des DeclareMiners treten vor allem auch dadurch auf, weil der verwendete LTL-Checker jede Regel separat und unabhängig voneinander prüft. Im Gegensatz zum Verfahren dieser Arbeit werden keine Gemeinsamkeiten der Regelkandidaten für eine effiziente Regelüberprüfung ausgenutzt. Des Weiteren werden alle möglichen Regelkandidaten generiert und überprüft. Parameterkombinationen, die im Ereignisprotokoll gar nicht zusammen in einer Prozessinstanz auftreten, werden nicht vorab gefiltert. Die resultierenden Modelle des ersten Ansatzes enthalten

Laufzeit-Vergleich

außerdem zahlreiche redundante Regeln, wodurch die Lesbarkeit des resultierenden Modells unnötig erschwert wird. Abbildung 46 zeigt ein Beispiel eines extrahierten Declare-Modells durch Anwendung des DeclareMiners in ProM [77].

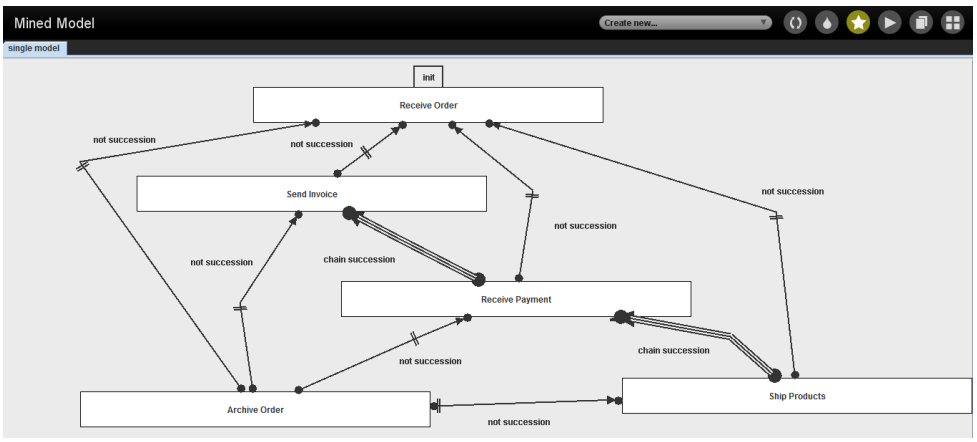


Abb. 46: Beispiel eines extrahierten Declare-Modells [77]

Erweiterungen des DeclareMiners

Es existieren eine Reihe von **Erweiterungen**, welche das ursprüngliche Verfahren mit Hinsicht auf einige der genannten Probleme verbessern. In [79] wird das **Laufzeit-Problem** des ersten Ansatzes betrachtet. Anstatt alle möglichen Regelkandidaten zu generieren und zu überprüfen, wird das Ereignisprotokoll voranalysiert und, ähnlich zu dem Verfahren in dieser Arbeit, lediglich häufig zusammen auftretende Parameter zur Kandidatenerzeugung verwendet. Auf diese Weise kann die Menge an zu überprüfenden Declare-Regeln stark reduziert und der Ansatz in der Praxis verwendet werden. Da Suchvorlagen des DeclareMiners lediglich Aktivitäten als Parameter verwenden, genügt es hier häufig zusammen auftretende Aktivitätenkombinationen abzuleiten. Durch Einbeziehung der organisatorischen Perspektive, wie in Kap. 5 dieser Arbeit, entstehen weitaus mehr Dimensionen. Hier müssen Aktivitäten/Identitäten- bzw. Aktivitäten/Gruppen-Kombinationen in die Analyse einbezogen werden.

Nachbearbeitung abgeleiteter Declare-Modelle

Das Problem der **Nachbearbeitung abgeleiteter Declare-Modelle**, d.h. das Entfernen redundanter Regeln, wird in [78] betrachtet. Dabei werden durch den Basisansatz abgeleitete Declare-Modelle hinsichtlich Redundanzen untersucht. Auf diese Weise können transitiv ableitbare, sowie Regeln, die bereits in anderen Regeln enthalten sind, identifiziert und entfernt werden. Anders als im Verfahren dieser Arbeit in Kap. 6 betreffen sämtliche Analysen die vorgegebene Menge an Declare-Regelvorlagen und somit lediglich die verhaltensorientierte Perspektive.

Erweiterungen für Analyse von Daten

Declare-Modelle können lediglich zeitliche Abhängigkeiten zwischen Aktivitäten darstellen, ohne andere Perspektiven zu berücksichtigen. Es fehlt daher ein Einblick in die teilweise komplexen Gründe für diese Abhängigkeiten. Die Forschung in [80] und [27] erweitern den DeclareMiner um **datenorientierte Aspekte**. Diese Erweiterungen können extrahieren, unter welchen (Daten-)Bedingungen eine Declare-Regel im Ereignisprotokoll gilt. Auch hier wird die

organisatorische Perspektive, d.h. komplexe Zuweisungsregeln bzw. der Einfluss auf den Prozessablauf, nicht betrachtet.

8.2.2 Effiziente Declare Mining-Verfahren

Unconstrained-Miner und Minerful

Aufgrund der Laufzeit-Probleme des ursprünglichen DeclareMiner, betrachteten mehrere Forschungsarbeiten, wie der *Minerful*-Ansatz nach DiCiccio [38], [45] und der *UnconstrainedMiner* nach Westergaard [139], vor allem die zeitliche Optimierung des Analyse-Vorgangs. Anstatt die Semantik von Declare-Regeln in LTL auszudrücken, verwenden diese Ansätze *reguläre Ausdrücke*. Eine *precedence(a,b)*-Regel mit dem LTL-Ausdruck $(\neg b)Wa$ kann beispielsweise durch den regulären Ausdruck $[\hat{}b] * (a. * b) * [\hat{}b]*$ abgebildet werden. Für jeden regulären Ausdruck existiert dann ein endlicher Automat mit mindestens einem Endzustand. Beide Verfahren erreichen extrem geringe Analyse-Laufzeiten. Auch diese Methoden erzeugen zuerst alle möglichen Regelkandidaten durch einen vollständigen Durchlauf des Ereignisprotokolls. Anschließend wird für jede Instanz überprüft, ob diese eine Sprache der ausgewählten Automaten darstellt. Ist eine bestimmte Instanz durch die Sprache eines Automaten nicht darstellbar, so ist diese Regel in dieser Instanz nicht erfüllt. Beide Verfahren leiten jedoch wiederum Declare-Modelle ab, wodurch die Defizite des ursprünglichen Verfahrens bestehen bleiben. Außerdem ist fraglich, ob im Fall einer Einbeziehung der organisatorischen Perspektive die Abbildung einer Regel in einem regulären Ausdruck möglich ist.

8.2.3 DecMiner auf Basis von SCIFF

SCIFF

Die Forschungsarbeiten in [37], [71], [89] stellen einen weiteren Ansatz zur Ableitung von regelbasierten Prozessmodellen dar, der ebenfalls auf Declare basiert. Als Zielsprache wird zuerst der SCIFF-Formalismus gewählt. SCIFF ist eine deklarative Sprache basierend auf abduktiver logischer Programmierung, die ursprünglich zur Spezifikation und Verifikation von Interaktionsprotokollen entwickelt wurde. Anschließend wird die abgeleitete Menge an SCIFF-Beschreibungen in ein Declare-Modell, d.h. ein DecSerFlow-Modell, übersetzt. Eine *precedence*-Regel zwischen den Aktivitäten a und b entspricht beispielsweise dem SCIFF-Ausdruck $H(b, T_b) \rightarrow E(a, T_a) \wedge T_a < T_b$.

Negative Beispiele werden benötigt

Der große Unterschied zu anderen regelbasierten Process Mining-Techniken ist, dass dieses Verfahren **negative Beispiele** benötigt. Die betrachteten Prozessinstanzen bzw. Spuren müssen vor der Analyse als konform bzw. nicht konform gekennzeichnet werden. Die Autoren motivieren dieses Szenario durch ein Beispiel. Eine Bank sei beispielsweise daran interessiert, in welchen Fällen Transaktionen fehlschlugen und in welchen sie normal, d.h. erfolgreich, abliefen. Mit diesem Mining-Verfahren können deshalb regalbasierte Modelle abgeleitet werden, die unterschiedliche Regeln in konformen und nicht konformen Fällen herausstellen. Das Problem ist, dass Prozessinstanzen im Allgemeinen

nicht als konform und nicht konform gekennzeichnet sind. Das Verfahren kann daher als Spezialfall des regelbasierten Process Mining angesehen werden, in dem mehr Information als gewöhnllich vorliegt.

8.2.4 Regelbasierte Konformitätsprüfung

Konformitätsprüfung prozeduraler Modelle

Neben Process Mining-Verfahren, die ein vollständiges, ausführbares Prozessmodell aus Ereignisprotokollen ableiten, existieren auch Techniken, die *vorgegebene* Regeln überprüfen, d.h. die Konformität eines Ereignisprotokolls und gegebener Regeln bzw. eines gegebenen Modells. Da die zu überprüfenden Regeln konkret vorgegeben werden müssen, können mit diesen Verfahren keine Modelle abgeleitet werden, sondern nur gegebene Modelle überprüft werden. Methoden zur **Konformitätsprüfung** (*Confirmance Checking*) verwenden ein Ereignisprotokoll und ein Modell und vergleichen das im Log aufgezeichnete mit dem modellierten Verhalten. Der Großteil der Methoden zur Konformitätsprüfung basiert dabei auf prozeduralen Prozessmodellierungssprachen wie Petri-Netzen [12], [134] und sind daher nicht direkt als verwandte Arbeiten zu betrachten.

Konformitätsprüfung regelbasierter Modelle

Die Konformität von *regelbasierten* Modellen und Ereignisprotokollen wurde in nur wenigen Forschungsarbeiten betrachtet. Einer der ersten Ansätze hierzu ist der bereits erwähnte *LTL-Checker* [3]. The LTL-Checker bietet verschiedene, in LTL definierte, konfigurierbare Regelvorlagen, mit denen bestimmte Zusammenhänge überprüft werden können, z.B. ob bestimmte Aktivitäten in einer Instanz durchgeführt wurden. Da sich Regeln im LTL-Checker nur auf konkrete Identitäten beziehen, können keine komplexen organisatorischen Zusammenhänge, wie beispielsweise rollenbasierte Zuweisungsregeln, überprüft werden. Das Verfahren in [19] betrachtet lediglich verhaltensorientierte Regeln. Der Ansatz in [43] betrachtet die Konformität von Logs und gegebenen Declare-Modellen. Dabei wird nicht nur dargestellt, welche Prozessinstanzen nicht konform zum Modell sind, sondern auch welchen Regeln diese widersprechen und wie oft diese verletzt sind. Auf diese Weise werden durch diesen Ansatz detaillierte Analysen ermöglicht. Da das Verfahren jedoch auf Declare-Modellen basiert, kann nur die verhaltensorientierte Perspektive betrachtet werden.

Überprüfung von Geschäftsregeln

In einigen Forschungsarbeiten wird dieser Anwendungsfall von Process Mining auch als Überprüfung von Geschäftsregeln bezeichnet (*engl. business rules checking*). Der neueste und mächtigste Ansatz zur Überprüfung von Geschäftsregeln ist das Verfahren nach Caron [33]–[35]. Analysten steht eine Menge von Regelvorlagen zur Verfügung, deren Parameter manuell gesetzt werden können. Da es sich um Regelvorlagen handelt, die alle Perspektiven der Prozessmodellierung einbeziehen, können auch die organisatorischen Muster dieser Arbeit formuliert und auf Basis eines Ereignisprotokolls überprüft werden. Wie bereits erwähnt, werden jedoch lediglich einzelne Regelinstanzen durch manuelle Spezifikation überprüft. Die Überprüfung einzelner Geschäftsregeln ist daher wesentlich weniger komplex und stellt nur einen Teilbereich des vorliegenden regelbasierten Process Mining-Verfahrens dar.

8.3 PROCESS MINING-VERFAHREN DER ORGANISATORISCHEN PERSPEKTIVE

Mining der organisatorischen Perspektive

Komplementär zu regelbasierten Process Mining-Verfahren existieren Techniken, welche die **organisatorische Perspektive von Prozessen** betrachten. Sie verwenden die häufig in Ereignisprotokollen zusätzlich aufgezeichnete Information über ausführende Personen bzw. Ressourcen. Existierende Process Mining-Methoden, die Ressourcen-Information in Logs analysieren, konzentrieren sich hauptsächlich darauf, ein zu Grunde liegendes Organisationsmodell oder verschiedene Arten von sozialen Netzwerken abzuleiten, welche die Interaktion von Personen darstellen. Des Weiteren existierenden außerdem Ansätze zur Analyse von organisatorischen Zuweisungsregeln. Zusätzlich wird der Einfluss von Ressourcen bzw. Personen auf die Prozessdurchlaufzeit analysiert [144]. Im folgenden Abschnitt werden die existierenden Ansätze hinsichtlich ihrer Analysefähigkeiten evaluiert. Als Rahmenwerk werden hierzu wiederum die organisatorischen Prozessmuster aus Abschnitt 4.2 verwendet. Zusätzlich wird evaluiert, ob eine Analyse perspektivenübergreifender Zusammenhänge möglich ist.

Organizational Miner

Der bereits in Abschnitt 2.4.2 eingeführte *OrganizationalMiner* nach Song [125] ermöglicht die automatisierte Ableitung von (einfachen) hierarchischen **Organisationsmodellen** aus Ereignisprotokollen. Der Ansatz verwendet agglomerative Clustering-Verfahren um konkrete Ressourcen zu Gruppen zusammenzufassen, wenn diese häufig identische Aktivitäten durchführen. Auch der Ansatz in [70] erzeugt Organisationsmodelle aus Ereignisprotokollen. Ähnlich zu [125] wird hier ein Ähnlichkeitsmaß verwendet, um Ressourcen zu Gruppen zu clustern. Als Datengrundlage wird dabei nur ein Ereignisprotokoll und kein Organisationsmodell benötigt. Beide Ansätze können zwar konkrete und rollenbasierte Zuweisungsregeln ableiten, komplexere Zuweisungsregeln werden jedoch nicht betrachtet.

Staff Assignment Miner

Der *StaffAssignmentMiner* [76], [107] ist ein Verfahren das **komplexere Zuweisungsregeln** von Ressourcen zu Aktivitäten betrachtet. Das Verfahren benötigt einerseits ein Ereignisprotokoll, andererseits ein Organisationsmodell zur Ableitung von Zuweisungsregeln. Mittels Entscheidungsbäumen (*engl. decision trees*) wird abgeleitet, welche Eigenschaften, Fähigkeiten und Rollen eine Ressource bzw. Person hat, um eine bestimmte Aktivität durchführen zu können. Auf diese Weise ist es möglich komplexe Zuweisungsregeln abzuleiten, die genau eine Aktivität betreffen, d.h. das fähigkeitsbasierte und das organisationsbasierte Zuweisungsmuster. Zuweisungsregeln, die mehrere Aktivitäten betreffen, wie eine Verknüpfung oder Trennung von Zuständigkeiten, oder eine organisationsbasierte Zuweisungsregel zwischen zwei oder mehr Aktivitäten, können nicht abgeleitet werden. Die Forschungsarbeiten in [24], [25], [73] sind in der Lage auch komplexe Zuweisungsregeln zwischen zwei Aktivitäten zu erkennen. Auf diese Weise ist es möglich eine personenbezogene als auch eine rollenbasierte Verknüpfung bzw. Trennung von Zuständigkeiten zwischen Aktivitäten abzuleiten. Organisationsbasierte Zuweisungsregeln, die mehr als

eine Aktivität betreffen, können nicht entdeckt werden. In beiden Ansätzen wird außerdem der Prozessablauf nicht mit in Analysen einbezogen, weshalb der Einfluss der organisatorischen Zusammenhänge auf den Kontrollfluss nicht analysiert werden kann.

Social Network Miner

Der *SocialNetworkMiner* [5] extrahiert ein *soziales Netzwerk*, d.h. einen Graphen, in denen jeder Knoten eine Ressource repräsentiert.

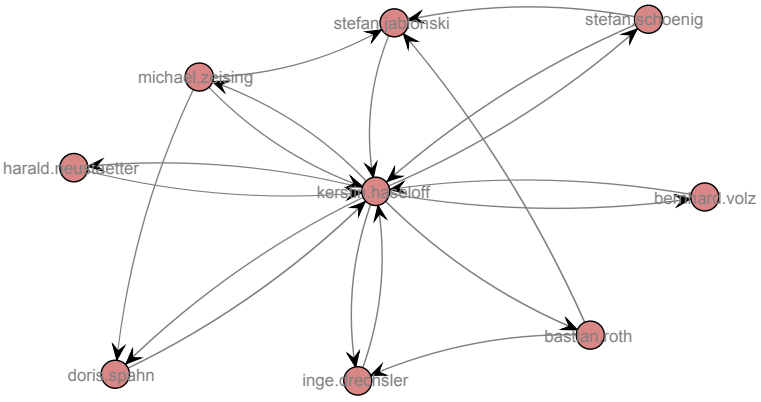


Abb. 47: Soziales Netzwerk "Übergabe von Arbeit"

Das Verfahren basiert auf vier verschiedenen Metriken: (1) Metrik auf Basis von Kausalität, (2) Metrik auf Basis gemeinsamer Fälle, (3) Metrik auf Basis gemeinsamer Aktivitäten und (4) Metrik auf Basis spezieller Ereignistypen. Auf Basis dieser Metriken werden Netzwerke erzeugt, welche Beziehungen und deren Gewicht zwischen Personen anzeigen. Ein Beispiel eines Netzwerks ist die "Übergabe von Arbeit". Innerhalb einer Prozessinstanz besteht die Beziehung "Übergabe von Arbeit" zwischen zwei Ressourcen i und j, wenn es zwei direkt aufeinanderfolgende Aktivitäten gibt, in denen die erste von i und die zweite von j durchgeführt wird. Da hier der Prozessablauf unter Einbeziehung von Ressourcen betrachtet wird, stellt die "Übergabe von Arbeitëinen perspektivenübergreifenden Zusammenhang dar. Dieser ist allerdings lediglich lokal, d.h. direkte aufeinanderfolgende Aktivitäten werden betrachtet. Abbildung 47 zeigt das resultierende soziale Netzwerk "Übergabe von Arbeit" nach Anwendung des Miners auf das Dienstreise-Ereignisprotokoll aus Kapitel 7.2 dieser Arbeit. Der Graph zeigt, welche Personen Aktivitäten direkt nacheinander durchgeführt haben. Der Miner interpretiert dies als eine Übergabe von Arbeit zwischen diesen Personen.

Einfluss von Ressourcen auf Prozess-Performance

Die Dissertation von Nakatumba [92], [93] beschäftigt sich eingehend mit dem Einfluss von Ressourcen auf die Prozessdurchführung mit einem starken Fokus auf Laufzeitanalyse. Hier werden jedoch keine Prozessmodelle abgeleitet, d.h. weder organisatorische Zuweisungsregeln noch perspektivenübergeifende Regeln. Der Ansatz ist daher eher als Ergänzung zu anderen Process Mining-Verfahren anzusehen.

Vergleich mit DpilMiner

Tabelle 18 gibt einen abschließenden Überblick über die Fähigkeiten der erläuterten organisatorischen Process Mining-Verfahren. Es wird ersichtlich, dass

Tabelle 18: Überblick organisatorische Mining-Verfahren und deren Fähigkeiten

Organisatorisches Muster	**Mining-Verfahren**	**DpilMiner**
Direkte Zuweisung	[25], [70], [107], [125]	✓
Rollenbasierte Zuweisung	[25], [70], [107], [125]	✓
Verzögerte Zuweisung		⊘
Trennung von Zuständigkeiten	[24], [25], [73]	✓
Fallbehandlung	[24], [25], [125]	✓
Verknüpfung von Zuständigkeiten	[24], [25], [73]	✓
Fähigkeitsbasierte Zuweisung	[107]	✓
Verlaufsbasierte Zuweisung		⊘
Organisationsbasierte Zuweisung	[107] (eine Aktivität)	✓ (mehrere Aktivitäten)
Perspektivenübergreifende Zusammenhänge	⊘	✓

sämtliche organisatorischen Muster, für die bislang mehrere verschiedene Miner benötigt werden, mit dem Verfahren dieser Arbeit abgeleitet werden können. Der große Vorteil zu existierenden Verfahren zeigt sich vor allem bei perspektivenübergreifenden Zusammenhängen und Zuweisungsregeln, die mehr als eine Aktivität gleichzeitig betreffen. Hier sind klassische Verfahren auf lokale Zusammenhänge eingeschränkt, wohingegen mit dem Ansatz der vorliegenden Arbeit beliebige Zusammenhänge abgeleitet werden können. Eine perspektivenübergreifende Betrachtung für benutzerdefinierte Zusammenhänge ist nur dem Ansatz dieser Arbeit möglich.

9 FAZIT UND AUSBLICK

INHALT

In diesem Kapitel werden Motivation, Ziele, Vorgehen und Ergebnisse der vorliegenden Dissertation zusammengefasst und Einschränkungen sowie zukünftige Forschungsthemen in diesem Bereich erläutert. Des Weiteren wird ein kurzer Einblick in die Publikationen des Autors im Bereich des Prozessmanagements gegeben.

9.1 ZUSAMMENFASSUNG

Process Discovery bezeichnet das Sammeln von Informationen über den Prozess und deren strukturierte Abbildung in einem Ist-Prozessmodell. Die Modellierungstätigkeit kann dabei erst beginnen, wenn genügend Informationen zusammengetragen wurden. Die Akquisition des Prozesswissens ist ein aufwändiges Unterfangen. Process Mining ist ein IT-gestütztes Verfahren zur Unterstützung und Automatisierung von Process Discovery.

Ausgangspunkt für die vorliegende Arbeit bildet die Beobachtung, dass zwei verschiedene Arten von Prozessen in Organisationen existieren, strikte und agile, personenbezogene Prozesse. Während klassische Process Mining-Techniken eher die Analyse strikter Prozesse adressiert, werden die eher agilen Prozesse nur unzureichend betrachtet. Agile, personenbezogene Prozesse, deren Abläufe sich nicht bereits im Voraus weitestgehend spezifizieren lassen, können durch prozedurale Modellierungssprachen weder adäquat abgebildet, noch durch existierende Process Mining-Verfahren zufriedenstellend analysiert werden. Es wird gezeigt, dass sich zur automatisierten Modellierung agiler, personenbezogener Prozesse ein regelbasierter Modellierungsansatz eignet, dass verpflichtende und empfohlene Handlungen unterschieden werden müssen und dass vor allem die organisatorische Perspektive und deren Einfluss auf den Prozessablauf untersucht werden muss. Im Rahmen der Arbeit wird daher ein Process Mining-Ansatz entwickelt, der in der Lage ist, regelbasierte Prozessmodelle aus Ereignisprotokollen abzuleiten und dabei die besonderen Anforderungen agiler, personenbezogener Prozesse berücksichtigt.

Als regelbasierte Zielsprache zur automatisierten Modellierung agiler, personenbezogener Prozesse wird im Rahmen der Arbeit die textuelle Sprache DPIL verwendet, in der die organisatorische Perspektive adäquat berücksichtigt wird und durch die eine perspektivenübergreifende Modellierung ermöglicht wird. Durch Regeln in verschiedenen Modalitäten wird eine Unterscheidung in verpflichtende und empfohlene Handlungen erlaubt. Zur automatisierten Ableitung von DPIL-Modellen wird ein Verfahren entworfen, in dem Zusammenhänge, die im gegebenen Ereignisprotokoll entdeckt werden sollen, durch Regel- bzw. Suchvorlagen in DPIL repräsentiert werden. Die Analyse einer bestimmten Regelvorlage kann als Suchanfrage an das gegebene Ereignisprotokoll betrachtet werden. Regelvorlagen können sowohl für komplexe organisatorische Zuweisungsregeln als auch für perspektivenübergreifende Zusammenhänge formuliert werden. Im eigentlichen Mining-Vorgang wird anhand des Logs überprüft, welchen Regeln der aufgezeichnete Prozess folgt. Regeln werden dabei unterschieden in (i) immer erfüllt, (i)) tendenziell erfüllt und (iii) häufig verletzt. Erfüllte und lediglich tendenziell erfüllte Regeln werden in unterschiedlicher Modalität in das resultierende, regelbasierte Prozessmodell übernommen. Tendenziell erfüllte Regeln stellen keine verpflichtenden jedoch empfohlene Aktionen dar. Durch eine effiziente Voranalyse der gegebenen Logs werden unnötige Berechnungen im Analyse-Vorgang vermieden. Zur Verbesserung der Verständlichkeit werden in einer Nachbearbeitungsphase überflüssige Zusammenhänge im abgeleiteten Modell entfernt.

Durch Anwendung der prototypischen Implementierung des vorgestellten Ansatzes auf ein Ereignisprotokoll eines agilen, personenbezogenen Prozesses, werden sowohl Funktionalität als auch Anwendbarkeit und Effizienz des Verfahrens belegt.

9.2 EINSCHRÄNKUNGEN DES ANSATZES

In diesem Abschnitt werden die Einschränkungen des vorliegenden Process Mining-Ansatzes aufgezeigt.

Einschränkungen von DPIL übertragen sich

- *Einschränkungen von DPIL übertragen sich auf DpilMiner*: Wie andere Process Mining-Verfahren auch, unterliegt der vorliegende Ansatz einer gewissen Darstellungsausrichtung (*representational bias*). Es können nur Muster und Regeln abgeleitet werden, welche sich auch durch die Sprache DPIL formulieren lassen. Auf diese Weise übertragen sich die Einschränkungen der Sprache direkt auf das Mining-Verfahren dieser Dissertation. Wie aus Tabelle 8 ersichtlich wird, kann das vorliegende Verfahren beispielsweise nicht das verlaufsbasierte Zuweisungsmuster erkennen. Dies liegt darin begründed, dass es mit der aktuellen Version von DPIL nicht möglich ist Regeln zu formulieren, die mehrere Prozessinstanzen betreffen (*Inter-Instanz Abhängigkeiten*). Es ist beispielsweise nicht möglich eine "Sachbearbeiter"-Regel zu formulieren und zu erkennen: Kundenanfra-

gen sollten vozugsweise von derjenigen Person bearbeitet werden, die auch bereits eine vorherige Anfrage dieses Kunden bearbeitet hat.

Viele Regeln in Ergebnismenge

- *Teilweise viele Regeln in Ergebnismenge*: Unter Umständen kann es vorkommen, dass bei einem Analysevorgang eine zu große, schwer verständliche Menge an Regeln extrahiert wird. Diese Gefahr besteht beispielsweise bei der Analyse der "Trennung von Zuständigkeiten" mit der *separate*-Suchvorlage. Eine *separate*-Regel wird zwischen allen Aktivitäten abgeleitet, die immer (verpflichtend) bzw. tendenziell (Empfehlung) von verschiedenen Personen durchgeführt werden. Wird nun ein Ereignisprotokoll eines Prozesses mit vielen verschiedenen beteiligten Personen analysiert, wobei jede Person nur wenige verschiedene Aktivitäten durchführt, so wird eine potentiell große Menge an *separate*-Regeln abgeleitet. Hier müssen evtl. Maßnahmen zur Reduzierung der Regelmenge gefunden werden, beispielsweise das manuelle Einschränken der möglichen beteiligten Aktivitäten durch den Analysten.

Nur häufigkeitsbasierte Empfehlungen

- *Einschränkung auf häufigkeitsbasierte Empfehlungen*: Im vorliegenden Ansatz werden verpflichtende und empfohlene Regeln unterschieden. Verpflichtende Regeln sind in nahezu allen Spuren des Ereignisprotokolls erfüllt und stellen daher den festen Rahmen dar, in dem sich der ausgeführte Prozess bewegt. Innerhalb dieses Rahmens können empfohlene Regeln Benutzern zusätzliche Unterstützung geben. In dieser Arbeit wird eine Regel als empfohlen klassifiziert, wenn sie zwar nicht in allen Spuren erfüllt ist, jedoch in einer genügend großen Anzahl an Instanzen. Eine Empfehlung ist daher häufigkeitsbasiert, d.h. eine Regel ist genau dann empfohlen, wenn der durch die Regel beschriebene Zusammenhang häufig im Ereignisprotokoll auftritt. Neben einer Häufigkeitsbetrachtung sind jedoch auch weitere, andersartige Bedingungen denkbar, bei deren Eintreten eine Regel als empfohlen klassifiziert werden kann.

9.3 ZUKÜNFTIGE FORSCHUNGSTHEMEN

In diesem Abschnitt werden zukünftige Forschungsthemen betrachtet, die sich aus der Forschung im Rahmen dieser Dissertation ergeben haben.

Weiterentwicklung von DPIL

- Ein zukünftiger Forschungsbereich, der sich aus der Forschung dieser Arbeit ergibt, ist die **Weiterentwicklung der Sprache DPIL**. Einerseits sollte die Sprache, wie bereits beschrieben, hinsichtlich der Abbildbarkeit von **Inter-Instanz-Abhängigkeiten** erweitert werden. Dies ermöglicht es Suchvorlagen in DPIL so zu formulieren, dass beispielsweise verlaufsbasierte Zuweisungsregeln abgeleitet werden können.

Graphische Notation für DPIL

- Des Weiteren ist eine **graphische Notation** von DPIL vorgesehen, was auch direkten Einfluss auf die Anwendbarkeit dies vorliegenden Mining-Ansatz hat. Durch eine graphische Notation sind resultierende Modelle

besser verständlich und auch Diskussionen mit Fachabteilungen (besser) möglich. Der Entwicklung einer graphischen Notation für DPIL sollte eine eingehende Recherche existierender graphischer Notation für regelbasierte Sprachen vorausgehen. Auf diese Weise könnten bestehende Konzepte für die graphische Notation von DPIL verwendet werden oder sogar eine vollständige Abbildung von DPIL auf eine existierende Notation erfolgen. Ein möglicher Kandidat hierfür ist die Notation *RALph* [31], welche sich vor allem mit der Darstellung von organisatorischen Mustern in Prozessen beschäftigt.

Mapping von DPIL auf RAL

- Wie in Abschnitt 3.1 beschrieben, existieren Ausdruckssprachen wie RAL [32] für prozedurale Modellierungssprachen, mit denen komplexe organisatorische Zuweisungsregeln formuliert werden können. In dieser Arbeit werden organisatorische Muster abgeleitet und in regelbasierten DPIL-Prozessmodellen abgebildet. Durch eine Abbildung von DPIL auf beispielsweise RAL könnte der vorliegende Ansatz auch zur Ableitung von organisatorischen **Zuweisungsregeln in prozeduralen Modellen** wie BPMN verwendet werden.

Nutzung von Regelhierarchien

- Die in Abschnitt 6.2 definierten organisatorischen Regelhierarchien können nicht nur für Process Mining-Verfahren von Nutzen sein. Ein zukünftiger Forschungsbereich könnte auch darin bestehen, organisatorische Regelhierarchien in Prozessmodellierungswerkzeugen zu verwenden, um Benutzer bei der Modellierung von möglichst übersichtlichen, redundanzfreien Modellen zu unterstützen.

Mining-Verfahren für CMMN

- Mit der Case Management Modeling and Notation (CMMN) wurde vor kurzem der erste Standard der regelbasierten Prozessmodellierung vorgestellt. Trotz einiger Schwächen bietet CMMN eine graphische Modellierungsnotation und wird in den kommenden Jahre an Bedeutung gewinnen. Ein Verfahren, das **CMMN-Modelle aus Ereignisprotokollen** extrahiert, ist daher sicherlich sinnvoll. Ein zukünftiges Forschungsthema ist daher die Evaluation und Abbildung von CMMN auf DPIL und die damit verbundene Möglichkeit, Suchvorlagen für die automatisierte Ableitung von CMMN-Modellen zu definieren.

Erweiterung auf andere Empfehlungsarten

- Ausgehend von der bereits erläuterten Einschränkung dieses Ansatzes, dass nur häufigkeitsbasierte Empfehlungen abgeleitet werden können, ist eine Erweiterung auf verschiedene **Arten von Bedingungen für empfohlene Regeln** denkbar. Eine Regel könnte beispielsweise dann empfohlen sein, wenn sie eine gegebene Kostenfunktion minimiert. Ein Beispiel hierfür wäre, dass die Anzahl an involvierten Mitarbeitern in einer Prozessinstanz zur Kostenreduktion minimiert werden soll. Zu diesem Zweck würde eine Fallbehandlung (*Case Handling*) empfohlen werden, d.h. alle Aktivitäten sollten von der gleichen Person durchgeführt werden, die bereits die erste Aktivität bearbeitet hat.

Mining-Verfahren für ortsbezogene Perspektive

- In [122] wurde die Sprache DPIL so erweitert, dass ortsbezogene Regeln ausgedrückt und ausgeführt werden können. So kann beispielsweise modelliert werden, dass eine Aktivität immer von der Person durchgeführt werden muss, die den kürzesten Weg zu den Koordinaten der Aufgabe hat. Diese Funktionalität ist verständlicherweise vor allem für ortsverteilte Prozesse, wie im Außendienst, interessant. Entsprechend ist auch eine Process Mining-Erweiterung denkbar, die aufgezeichnete Prozessabläufe aus einer **ortsbezogenen Perspektive** betrachet. Mögliche Analysen könnten unter Einbeziehung von GPS-Koordinaten in Ereignisprotokollen ableiten, auf Basis welcher ortsbezogenen Bedinungen Aktivitäten an Mitarbeiter zugewiesen wurden.

9.4 PUBLIKATIONEN DES AUTORS

Der Großteil der Ergebnisse dieser Dissertation wurden bereits auf wissenschaftlichen Konferenzen oder in Zeitschriften veröffentlicht und/oder in industriellen Anwendungsfällen im Rahmen des KpPQ-Projektes evaluiert. Des Weiteren wurden auf Basis weiterer Forschungsaktivitäten im Bereich des Prozessmanagements, welche nicht Vordergründig der Thematik dieser Arbeit zuzuordnen sind, weitere wissenschaftliche Artikel publiziert.

KpPQ-Projekt

Abbildung 48 gibt einen Überblick über sämtliche Publikationen die im Laufe dieser Arbeit angefertigt wurden. Diese Publikationen werden nach deren Status, Typ und Thema klassifiziert. Die verschiedenen Typen (Workshop-, Konferenz- oder Journal-Artikel) sind in Klammern hinter dem Namen der jeweiligen Konferenz bzw. dem Journal angegeben. Die Publikationen können fünf verschiedenen Themen zugeordnet werden: (i) die grüne Linie zeigt Publikationen die sich dem (regelbasierten) Process Mining zuordnen lassen; (ii) rote Veröffentlichungen behandeln die Vorverarbeitung von Ereignisprotokollen zur Ableitung von häufigen Mustern; (iii) die blaue Linie vereint sämtliche Publikationen die sich mit der regelbasierten Sprache DPIL beschäftigen; (iv) die Veröffentlichungen, die sich auf die Akquisition von Ereignisprotokollen beziehen, sind gelb dargestellt; (v) der braune Bereich zeigt Publikationen in anderen Forschungsbereichen des Prozessmanagements, wie dem automatisierten Auffinden von Überschneidungen in Prozessmodellen (*engl. process model similarity matching*) oder der Transformation von Prozessmodellen in eine Checklisten-Darstellung (*engl. model transformation to process checklists*). Schwarze Kreise zeigen an, dass eine Publikation bereits veröffentlicht ist. Grüne Kreise stellen Publikationen dar, die angenommen aber noch nicht veröffentlicht sind. Rote Kreise hingegen stellen eingereichte Artikel dar, für die zum Zeitpunkt der Abgabe dieser Arbeit noch kein Ergebnis vorliegt.

Publikationen des Autors

Die Veröffentlichungen im Bereich **regelbasiertes Process Mining** (*engl. rule based, declarative process mining*) befassen sich mit den verschiedenen Aspekten des Process Mining. Während in den ersten Veröffentlichungen eher der Bedarf

Veröffentlichungen im Bereich Process Mining

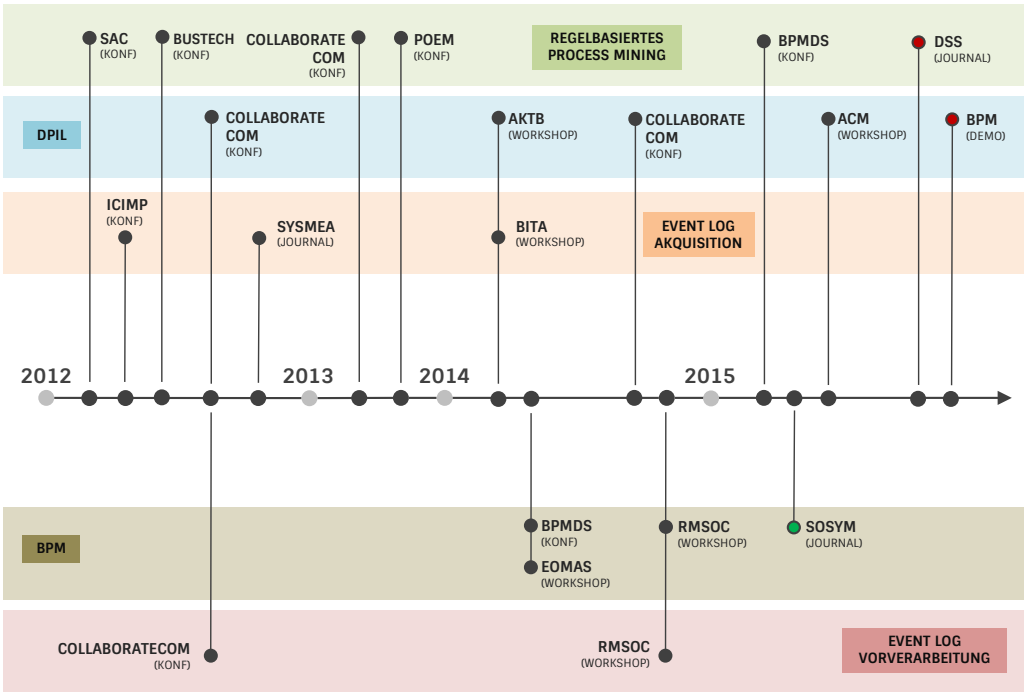


Abb. 48: Übersicht über die Publikationen des Autors

für regelbasierte Verfahren erläutert wird, beschreiben die späteren Veröffentlichungen eher die konzeptuellen Details und technischen Umsetzungen.

- *Dynamic Guidance Enhancement in Workflow Management Systems* (27th Symposium On Applied Computing, SAC 2012) [60]: In dieser Veröffentlichung betrachten wir den Bedarf und erste konzeptuelle Ansätze für Empfehlungen in agilen Prozessmanagementsystemen auf Basis von regelbasiertem Process Mining.

- *Discovering Cross-Perspective Semantic Definitions from Process Execution Logs* (Second International Conference on Business Intelligence and Technology, BUSTECH 2012) [119]: Diese Publikation führt zum ersten Mal den Begriff perspektivenübergreifender Zusammenhänge ein. Es wird das grundsätzliche Verfahren zur Ableitung von Regeln auf Basis von Regelvorlagen vorgestellt.

- *Supporting Collaborative Work by Learning Process Models and Patterns from Cases* (International Conference on Collaborative Computing, CollaborateCom 2013) [120]: In dieser Arbeit wird regelbasiertes Process Mining verwendet um Modelle von Case Management-Prozessen abzuleiten.

- *Comprehensive Business Process Management through Observation and Navigation* (Working Conference on the Practice of Enterprise Modeling, PoEM 2013) [121]: Diese Publikation erläutert die Integration von regelbasiertem Process Mining und einem entsprechenden regelbasierten Prozessausführungssystem.

- *Mining the Organisational Perspective in Agile Business Processes* (Working Conference on Business Process Modeling, Development, and Support, BPMDS 2015) [112]: In dieser Arbeit fokussieren wir die Anwendung von regelbasiertem Process Mining im Kontext der organisatorischen Perspektive von Prozessen. Außerdem wird der Ansatz mit einem realen Ereignisprotokoll evaluiert.
- *A Framework for Mining the Organisational Perspective in Agile Business Processes* (Decision Support Systems): Dieser Journal-Artikel stellt eine Zusammenfassung der vorliegenden Dissertation dar und enthält im Wesentlichen alle Inhalte, die auch in dieser Arbeit vorgestellt werden.

Publikationen über DPIL

Die Publikationen im Bereich **DPIL** *Declarative Process Intermediate Language* geben vor allem Einblick in Entwicklung, Umsetzung und IT-gestützte Ausführung der regelbasierten Sprache DPIL, die als Zielsprache des Mining-Verfahrens dieser Arbeit verwendet wird.

- *Improving Collaborative Business Process Execution by Traceability and Expressiveness* (International Conference on Collaborative Computing, CollaborateCom 2012) [142]: In dieser Publikation wird das Ausführungsprinzip für eine allgemeine, regelbasierte Sprache auf Basis der Prädikatenlogik 1. Stufe erläutert. Diese Arbeit legt den Grundstein für die Entwicklung von DPIL.
- *Towards Location-Aware Declarative Business Process Management* (Workshop on Applications of Knowledge-Based Technologies in Business, AKTB 2014) [122]: In dieser Publikation wird die Sprache DPIL eingeführt. Die Erweiterbarkeit der Sprache wird anhand von regelbasierten Konstrukten zur Abbildung von ortsabhängigen Zusammenhängen erläutert.
- *Towards a Common Platform for the Support of Routine and Agile Business Processes* (International Conference on Collaborative Computing, CollaborateCom 2014) [143]: In dieser Veröffentlichung wird die Ausdrucksstärke von DPIL hinsichtlich der häufigen Prozessmuster (Workflow Patterns) evaluiert und gegen klassische Sprachen wie BPMN abgegrenzt.
- *Comparing Declarative Process Modelling Languages from the Organisational Perspective* (4th International Workshop on Adaptive Case Management and other non-workflow approaches to BPM, AdaptiveCM 2015) [114]: Diese Veröffentlichung stellt eine umfassende Evaluation existierender deklarativer Prozessmodellierungssprachen dar. Insbesondere werden die Fähigkeiten zur Abbildung organisatorischer Prozessmuster betrachtet. Es werden Schwachstellen aufgezeigt und mögliche Lösungsvorschläge gegeben.
- *The DPIL Framework: Tool Support for Agile and Resource-Aware Business Processes* (BPM 2015 Demos): In dieser Veröffentlichung im Demo-Track der BPM-Konferenz wird das vollständige DPIL-Rahmenwerk bestehend aus

Modellierungseditor, Ausführungsumgebung und Mining-Modul vorgestellt.

Allgemeine BPM-Publikationen

Im Bereich **BPM** (*Business Process Management*) finden sich alle Publikationen, die nicht direkt dem Themenbereich dieser Arbeit zugeordnet werden können. Sie geben Einblick in die weiteren Forschungsbereiche des Prozessmanagements, in denen der Autor wissenschaftliche Publikationen verzeichnen kann.

- *Enhancing Feasibility of Human-driven Processes by Transforming Process Models to Process Checklists* (Working Conference on Business Process Modeling, Development, and Support, BPMDS 2014) [23]: Diese Publikation erläutert den Bedarf und die Motivation für die papierbasierte Unterstützung von Prozessen durch prozessbasierte Checklisten. Es wird ferner eine algorithmische Abbildung von einfachen BPMN-Elementen auf Checklisten gegeben.

- *Towards Multi-Perspective Process Model Similarity Matching* (Workshop on Enterprise and Organizational Modeling and Simulation, EOMAS 2014) [22]: Diese Publikation betrachtet den Ähnlichkeitsvergleich von gegebenen Prozessmodellen unter Einbeziehung der verschiedenen Perspektiven der Prozessmodellierung im Allgemeinen.

- *Resource-Aware Process Model Similarity Matching* (Workshop on Resource Management in Service-Oriented Computing, RMSOC 2014) [21]: Die Ähnlichkeitsuntersuchung wird in dieser Veröffentlichung im Hinblick auf verschiedene organisatorische Gegebenheiten erweitert. Auf diese Weise wird ein Ähnlichkeitswert von Prozessmodellen mit Fokus auf die organisatorische Perspektive des Prozesses berechnet.

- *The Process Checklist - Simple Establishment of Execution Support for Human-driven Processes* (Software and Systems Modeling): Dieser Journal-Artikel stellt eine Erweiterung der Konferenz-Veröffentlichung über Checklisten dar. Hier werden weitere Konzepte sowie eine umfassende Evaluation des Checklisten-Ansatzes vorgestellt.

Publikationen im Bereich Log Akquisition

Der Bereich **Event Log Akquisition** beinhaltet Veröffentlichungen, welche sich mit der Erzeugung von analysierbaren, digitalen Ereignisprotokollen befassen.

- *Process Discovery and Guidance Applications of Manually Generated Logs* (Seventh Conference on Internet Monitoring and Protection, ICIMP 2012) [113]: In dieser Veröffentlichung wird die regelfreie Ausführung von Prozessen betrachtet und als *Process Observation* bezeichnet. Auf diese Weise werden Ereignisprotokolle hoher Qualität erzeugt, die von Process Mining-Techniken analysiert werden können.

- *Process Observation as Support for Evolutionary Process Engineering* (Advances in Systems and Measurements) [115]: In diesem Journal-Artikel wird das Prinzip der Process Observation angewendet um die Prozessunterstützung evolutionär zu verbessern.
- *A Framework for Reasonable Support of Process Compliance Management* (Workshop on Business and IT Alignment (BITA 2014) [123]: Auch diese Veröffentlichung befasst sich mit der Anwendung der regelfreien Prozessdurchführung zur evolutionären Unterstützung von Prozessen. Der Fokus liegt hierbei auf dem Nachweis der Einhaltung von Rahmenbedingung.

Veröffentlichungen zur Log-Voranalyse

Die Publikationen im Bereich der **Event Log Vorverarbeitung** verwenden Ansätze zur Ableitung von häufigen Mustern zur Extraktion von Parameterkombinationen für das regelbasierte Process Mining.

- *Adapting Association Rule Mining to Discover Patterns of Collaboration in Process Logs* (International Conference on Collaborative Computing, CollaborateCom 2012) [118]: Diese Publikation erläutert die Anwendung von Association Rule Mining-Verfahren zur Ableitung von perspektivenübergeifenden Mustern.
- *Supporting Rule-based Process Mining by User-Guided Discovery of Resource-Aware Frequent Patterns* (Workshop on Resource Management in Service-Oriented Computing, RMSOC 2014) [117]: In dieser Publikation wird die Ableitung von häufigen Mustern als Ausgangspunkt für regelbasierte Process Mining-Verfahren behandelt. Der Fokus liegt dabei vor allem auf Mustern, welche Ressourcen, d.h. die organisatorische Perspektive, einbeziehen.

A ANHANG

INHALT

A.1 DIENSTREISEABWICKLUNGSPROZESS

In diesem Abschnitt sind in Listing 28 zwei vollständige Spuren des aufgezeichneten Ereignisprotokolls des Dienstreiseabwicklungsprozesses sowie das vollständige extrahierte DPIL-Prozessmodell abgebildet.

A.1.1 Ausschnitt des Ereignisprotokolls

Listing 28: Ereignisprotokolls des Dienstreiseprozesses

```
<log>
<trace>
<string key="concept:name" value="SS_RigaLettland112013"/>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-06T14:58:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-06T15:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SJ"/>
        <date key="time:timestamp" value="2013-06-07T10:22:00.000+01:00"/>
```

```
        <string key="concept:name" value="Antrag intern genehmigen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SJ"/>
        <date key="time:timestamp" value="2013-06-07T10:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag intern genehmigen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-08T09:02:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (intern)"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-08T09:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (intern)"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-06-08T14:45:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (Verwaltung)"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-06-08T15:56:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (Verwaltung)"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-09T08:32:00.000+01:00"/>
        <string key="concept:name" value="Rueckmeldung bearbeiten"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-09T08:36:00.000+01:00"/>
        <string key="concept:name" value="Rueckmeldung bearbeiten"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-09T10:33:00.000+01:00"/>
        <string key="concept:name" value="Flug buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
```

```
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-09T10:58:00.000+01:00"/>
        <string key="concept:name" value="Flug buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-09T11:33:00.000+01:00"/>
        <string key="concept:name" value="Transfer buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="20130611-09T11:38:00.000+01:00"/>
        <string key="concept:name" value="Transfer buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-09T12:33:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SS"/>
        <date key="time:timestamp" value="2013-06-09T12:38:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-10T08:12:00.000+01:00"/>
        <string key="concept:name" value="Dienstreise archivieren"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-06-10T08:36:00.000+01:00"/>
        <string key="concept:name" value="Dienstreise archivieren"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
</trace>
<trace>
<string key="concept:name" value="MZ_MUC112013"/>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-06T14:58:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
```

```
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-06T15:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag stellen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="SJ"/>
        <date key="time:timestamp" value="2013-11-07T10:22:00.000+01:00"/>
        <string key="concept:name" value="Antrag intern genehmigen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="SJ"/>
        <date key="time:timestamp" value="2013-11-07T10:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag intern genehmigen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-08T09:02:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (intern)"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-08T09:31:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (intern)"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-11-08T14:45:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (Verwaltung)"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="DS"/>
        <date key="time:timestamp" value="2013-11-08T15:56:00.000+01:00"/>
        <string key="concept:name" value="Antrag pruefen (Verwaltung)"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-09T08:32:00.000+01:00"/>
        <string key="concept:name" value="Rueckmeldung bearbeiten"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-09T08:36:00.000+01:00"/>
```

```
        <string key="concept:name" value="Rueckmeldung bearbeiten"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T10:31:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T10:32:00.000+01:00"/>
        <string key="concept:name" value="Unterkunft buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T10:33:00.000+01:00"/>
        <string key="concept:name" value="Zug buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T10:58:00.000+01:00"/>
        <string key="concept:name" value="Zug buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T11:33:00.000+01:00"/>
        <string key="concept:name" value="Transfer buchen"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="MZ"/>
        <date key="time:timestamp" value="2013-11-09T11:38:00.000+01:00"/>
        <string key="concept:name" value="Transfer buchen"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-10T08:12:00.000+01:00"/>
        <string key="concept:name" value="Dienstreise archivieren"/>
        <string key="lifecycle:transition" value="start"/>
</event>
<event>
        <string key="org:resource" value="KH"/>
        <date key="time:timestamp" value="2013-11-10T08:36:00.000+01:00"/>
        <string key="concept:name" value="Dienstreise archivieren"/>
        <string key="lifecycle:transition" value="complete"/>
</event>
```

```
</trace>
```

A.1.2 Petrinetz Prozessmodell des α-Algorithmus

In Abb. 49 ist das vollständige, extrahierte Prozessmodell des α-Algorithmus dargestellt.

A.1.3 Vollständige DPIL-Prozessmodelle

In diesem Abschnitt werden die vollständigen, mit dem DpilMiner extrahierten, DPIL-Prozessmodell des Dienstreiseabwicklungsprozesses aus Abschnitt 7.2 visualisiert. Der DpilMiner wird dabei mit verschiedenen benutzerdefinierten Schwellenwerten in der Vorverarbeitungsphase und während des eigentlichen Mining-Vorgangs konfiguriert. Vor jeder Regel ist in Klammern angegeben, wie diese während des Evaluationsworkshops klassifiziert worden ist. Die Regeln, die mit *(FN)* gekennzeichnet sind, stellen fehlende Regeln dar, die nicht abgeleitet wurden. Diese sind im Evaluationsworkshop jedoch als notwendig und relevant erachtet worden.

Listing 29 zeigt das DPIL-Modell, das durch das Verfahren mit $\mathit{minSupp} = 10\%$ und $\mathit{minConf}_S = 85\%$ abgeleitet wurde.

Listing 29: DPIL-Modell M2 des Dienstreiseprozesses

```
%Identitaeten
use identity SS
use identity SJ
use identity KH
use identity DS
use identity MZ
use identity BR
use identity LA
use identity MB
use identity ID
use identity HN

%Gruppen bzw. Rollen
use group Professor
use group Sekretariat
use group Administration
use group Doktorand

%Beziehungstypen
use relationtype hasRole
use relationtype supervisor

process Dienstreiseabwicklung {
    %Funktionale Perspektive
    task Antragstellen "Antrag stellen"
    task Antraginterngenehmigen "Antrag intern genehmigen"
    task Antragpruefen(intern) "Antrag pruefen (intern)"
```

```
task Antragpruefen(Verwaltung) "Antrag pruefen (Verwaltung)"
task Rueckmeldungbearbeiten "Rueckmeldung bearbeiten"
task Flugbuchen "Flug buchen"
task Transferbuchen "Transfer buchen"
task Unterkunftbuchen "Unterkunft buchen"
task Dienstreisearchivieren "Dienstreise archivieren"

(Missing) task Zugbuchen "Zug buchen"

%Verhaltensorientierte Perspektive
(TP) ensure sequence(Antragstellen,Antraginterngenehmigen)
(TP) ensure sequence(Antragstellen,Antragpruefen(intern))
(TP) ensure sequence(Antragpruefen(intern),Antragpruefen(Verwaltung))
(TP) ensure sequence(Antragpruefen(Verwaltung),Rueckmeldungbearbeiten)
(TP) ensure sequence(Rueckmeldungbearbeiten,Dienstreisearchivieren)

(FP) ensure sequence(Flugbuchen,Transferbuchen)

(TP) advice sequence(Antragstellen,Flugbuchen)
(TP) advice sequence(Antragstellen,Transferbuchen)
(TP) advice sequence(Antragstellen,Unterkunftbuchen)
(TP) advice sequence(Antragpruefen(intern),Flugbuchen)
(TP) advice sequence(Antragpruefen(intern),Transferbuchen)
(TP) advice sequence(Antragpruefen(intern),Unterkunftbuchen)

(FP) advice sequence(Unterkunftbuchen,Transferbuchen)
(FP) advice sequence(Unterkunftbuchen,Dienstreisearchivieren)

(FN) advice sequence(Antragstellen,Zugbuchen)
(FN) advice sequence(Antragpruefen(intern),Zugbuchen)

%Organisatorische Perspektive
(TP) ensure direct(Antragpruefen(intern), KH)
(TP) ensure direct(Rueckmeldungbearbeiten, KH)
(TP) ensure direct(Dienstreisearchivieren, KH)
(TP) ensure direct(Antraginterngenehmigen, SJ)
(TP) ensure role(Antragpruefen(Verwaltung), Administration)
(TP) ensure binding(Antragstellen,Flugbuchen)
(TP) ensure binding(Antragstellen,Transferbuchen)
(TP) ensure binding(Antragstellen,Unterkunftbuchen)

(FP) ensure binding(Antragpruefen(intern),Rueckmeldungbearbeiten)
(FP) ensure binding(Antragpruefen(intern),Dienstreisearchivieren)

(TP) ensure orgDistM(Antraginterngenehmigen,Antragstellen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Flugbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Transferbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Unterkunftbuchen,supervisor)

(FN) ensure binding(Antragstellen,Zugbuchen)
(FN) ensure orgDistM(Antraginterngenehmigen,Zugbuchen,supervisor)
```

```
    %Perspektivenuebergreifende Regeln
    (TP) ensure roleSequence(Antraginterngenehmigen,Transferbuchen,Doktorand)
    (TP) ensure roleSequence(Antraginterngenehmigen,Flugbuchen,Doktorand)
    (TP) ensure roleSequence(Antraginterngenehmigen,Unterkunftbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Flugbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Transferbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Unterkunftbuchen,Doktorand)

    (FN) ensure roleSequence(Antraginterngenehmigen,Zugbuchen,Doktorand)
    (FN) ensure roleSequence(Antragpruefen(intern),Zugbuchen,Doktorand)

    %Abschlussbedingungen
    milestone "Done":
        (TP) complete(of Antrag stellen) and
        (TP) complete(of Antrag pruefen (intern)) and
        (TP) complete(of Antrag pruefen (Verwaltung) and
        (TP) complete(of Rueckmeldung bearbeiten) and
        (TP) complete(of Dienstreise archivieren)
}
```

Listing 30 zeigt das DPIL-Modell M3, das durch das Verfahren mit $minSupp = 10\%$ und einem Confidence-Wert von $minConf_S = 90\%$ abgeleitet wurde.

Listing 30: DPIL-Modell M3 des Dienstreiseprozesses

```
%Identitaeten
use identity SS
use identity SJ
use identity KH
use identity DS
use identity MZ
use identity BR
use identity LA
use identity MB
use identity ID
use identity HN

%Gruppen bzw. Rollen
use group Professor
use group Sekretariat
use group Administration
use group Doktorand

%Beziehungstypen
use relationtype hasRole
use relationtype supervisor

process Dienstreiseabwicklung {
    %Funktionale Perspektive
    task Antragstellen "Antrag stellen"
    task Antraginterngenehmigen "Antrag intern genehmigen"
```

```
task Antragpruefen(intern) "Antrag pruefen (intern)"
task Antragpruefen(Verwaltung) "Antrag pruefen (Verwaltung)"
task Rueckmeldungbearbeiten "Rueckmeldung bearbeiten"
task Flugbuchen "Flug buchen"
task Transferbuchen "Transfer buchen"
task Unterkunftbuchen "Unterkunft buchen"
task Dienstreisearchivieren "Dienstreise archivieren"

(Missing) task Zugbuchen "Zug buchen"

%Verhaltensorientierte Perspektive
(TP) ensure sequence(Antragstellen,Antraginterngenehmigen)
(TP) ensure sequence(Antragstellen,Antragpruefen(intern))
(TP) ensure sequence(Antragpruefen(intern),Antragpruefen(Verwaltung))
(TP) ensure sequence(Antragpruefen(Verwaltung),Rueckmeldungbearbeiten)
(TP) ensure sequence(Rueckmeldungbearbeiten,Dienstreisearchivieren)
(FP) ensure sequence(Flugbuchen,Transferbuchen)

(FN) advice sequence(Antragstellen,Flugbuchen)
(FN) advice sequence(Antragstellen,Transferbuchen)
(FN) advice sequence(Antragstellen,Unterkunftbuchen)
(FN) advice sequence(Antragpruefen(intern),Flugbuchen)
(FN) advice sequence(Antragpruefen(intern),Transferbuchen)
(FN) advice sequence(Antragpruefen(intern),Unterkunftbuchen)
(FN) advice sequence(Antragstellen,Zugbuchen)
(FN) advice sequence(Antragpruefen(intern),Zugbuchen)

%Organisatorische Perspektive
(TP) ensure direct(Antragpruefen(intern), KH)
(TP) ensure direct(Rueckmeldungbearbeiten, KH)
(TP) ensure direct(Dienstreisearchivieren, KH)
(TP) ensure direct(Antraginterngenehmigen, SJ)
(TP) ensure role(Antragpruefen(Verwaltung), Administration)
(TP) ensure binding(Antragstellen,Flugbuchen)
(TP) ensure binding(Antragstellen,Transferbuchen)
(TP) ensure binding(Antragstellen,Unterkunftbuchen)

(FP) ensure binding(Antragpruefen(intern),Rueckmeldungbearbeiten)
(FP) ensure binding(Antragpruefen(intern),Dienstreisearchivieren)

(TP) ensure orgDistM(Antraginterngenehmigen,Antragstellen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Flugbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Transferbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Unterkunftbuchen,supervisor)

(FN) ensure binding(Antragstellen,Zugbuchen)
(FN) ensure orgDistM(Antraginterngenehmigen,Zugbuchen,supervisor)

%Perspektivenuebergreifende Regeln
(TP) ensure roleSequence(Antraginterngenehmigen,Transferbuchen,Doktorand)
(TP) ensure roleSequence(Antraginterngenehmigen,Flugbuchen,Doktorand)
```

```
    (TP) ensure roleSequence(Antraginterngenehmigen,Unterkunftbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Flugbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Transferbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Unterkunftbuchen,Doktorand)

    (FN) ensure roleSequence(Antraginterngenehmigen,Zugbuchen,Doktorand)
    (FN) ensure roleSequence(Antragpruefen(intern),Zugbuchen,Doktorand)

    %Abschlussbedingungen
    milestone "Done":
        (TP) complete(of Antrag stellen) and
        (TP) complete(of Antrag pruefen (intern)) and
        (TP) complete(of Antrag pruefen (Verwaltung) and
        (TP) complete(of Rueckmeldung bearbeiten) and
        (TP) complete(of Dienstreise archivieren)
}
```

Listing 31 zeigt das DPIL-Modell M1, das durch das Verfahren ohne Vorverarbeitungsphase und mit einem Confidence-Wert von $minConf_S = 85\%$ abgeleitet wurde.

Listing 31: DPIL-Modell M1 des Dienstreiseprozesses

```
%Identitaeten
use identity SS
use identity SJ
use identity KH
use identity DS
use identity MZ
use identity BR
use identity LA
use identity MB
use identity ID
use identity HN

%Gruppen bzw. Rollen
use group Professor
use group Sekretariat
use group Administration
use group Doktorand

%Beziehungstypen
use relationtype hasRole
use relationtype supervisor

process Dienstreiseabwicklung {
    %Funktionale Perspektive
    task Antragstellen "Antrag stellen"
    task Antraginterngenehmigen "Antrag intern genehmigen"
    task Antragpruefen(intern) "Antrag pruefen (intern)"
    task Antragpruefen(Verwaltung) "Antrag pruefen (Verwaltung)"
    task Rueckmeldungbearbeiten "Rueckmeldung bearbeiten"
```

```
task Flugbuchen "Flug buchen"
task Transferbuchen "Transfer buchen"
task Unterkunftbuchen "Unterkunft buchen"
task Dienstreisearchivieren "Dienstreise archivieren"
task Zugbuchen "Zug buchen"

%Verhaltensorientierte Perspektive
(TP) ensure sequence(Antragstellen,Antraginterngenehmigen)
(TP) ensure sequence(Antragstellen,Antragpruefen(intern))
(TP) ensure sequence(Antragpruefen(intern),Antragpruefen(Verwaltung))
(TP) ensure sequence(Antragpruefen(Verwaltung),Rueckmeldungbearbeiten)
(TP) ensure sequence(Rueckmeldungbearbeiten,Dienstreisearchivieren)
(FP) ensure sequence(Flugbuchen,Transferbuchen)

(TP) advice sequence(Antragstellen,Flugbuchen)
(TP) advice sequence(Antragstellen,Transferbuchen)
(TP) advice sequence(Antragstellen,Unterkunftbuchen)
(TP) advice sequence(Antragstellen,Zugbuchen)
(TP) advice sequence(Antragpruefen(intern),Flugbuchen)
(TP) advice sequence(Antragpruefen(intern),Transferbuchen)
(TP) advice sequence(Antragpruefen(intern),Unterkunftbuchen)
(TP) advice sequence(Antragpruefen(intern),Zugbuchen)

(FP) advice sequence(Unterkunftbuchen,Transferbuchen)
(FP) advice sequence(Unterkunftbuchen,Dienstreisearchivieren)

%Organisatorische Perspektive
(TP) ensure direct(Antragpruefen(intern), KH)
(TP) ensure direct(Rueckmeldungbearbeiten, KH)
(TP) ensure direct(Dienstreisearchivieren, KH)
(TP) ensure direct(Antraginterngenehmigen, SJ)
(TP) ensure role(Antragpruefen(Verwaltung), Administration)

(TP) ensure binding(Antragstellen,Flugbuchen)
(TP) ensure binding(Antragstellen,Transferbuchen)
(TP) ensure binding(Antragstellen,Unterkunftbuchen)
(TP) ensure binding(Antragstellen,Zugbuchen)

(FP) ensure binding(Antragpruefen(intern),Rueckmeldungbearbeiten)
(FP) ensure binding(Antragpruefen(intern),Dienstreisearchivieren)

(TP) ensure orgDistM(Antraginterngenehmigen,Antragstellen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Flugbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Transferbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Unterkunftbuchen,supervisor)
(TP) ensure orgDistM(Antraginterngenehmigen,Zugbuchen,supervisor)

%Perspektivenuebergreifende Regeln
(TP) ensure roleSequence(Antraginterngenehmigen,Transferbuchen,Doktorand)
(TP) ensure roleSequence(Antraginterngenehmigen,Flugbuchen,Doktorand)
(TP) ensure roleSequence(Antraginterngenehmigen,Unterkunftbuchen,Doktorand)
```

```
    (TP) ensure roleSequence(Antragpruefen(intern),Flugbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Transferbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Unterkunftbuchen,Doktorand)
    (TP) ensure roleSequence(Antraginterngenehmigen,Zugbuchen,Doktorand)
    (TP) ensure roleSequence(Antragpruefen(intern),Zugbuchen,Doktorand)

    (FP) ensure roleSequence(Antragpruefen(Verwaltung),Zugbuchen,Doktorand)
    (FP) ensure roleSequence(Rueckmeldungbearbeiten,Zugbuchen,Doktorand)

    %Abschlussbedingungen
    milestone "Done":
        (TP) complete(of Antrag stellen) and
        (TP) complete(of Antrag pruefen (intern)) and
        (TP) complete(of Antrag pruefen (Verwaltung) and
        (TP) complete(of Rueckmeldung bearbeiten) and
        (TP) complete(of Dienstreise archivieren)
}
```

A.2 KLINISCHER PROZESS

A.2.1 Prozessmodelle klassischer Verfahren

In Abb. 50 ist das vollständige, extrahierte Prozessmodell des α-Algorithmus (*ProM Version 6.3* Implementierung) dargestellt.

In Abb. 51 ist das vollständige, extrahierte Prozessmodell des Fuzzy Miners (*ProM Version 6.3* Implementierung) dargestellt.

In Abb. 52 ist das vollständige, extrahierte Prozessmodell des Fuzzy Miners (*ProM Version 6.3* Implementierung) dargestellt.

A.2.2 Vollständige DPIL-Prozessmodelle

In diesem Abschnitt werden die vollständigen, mit dem DpilMiner extrahierten, DPIL-Prozessmodelle des klinischen Prozesses aus Abschnitt 7.3 visualisiert. Listing 32 zeigt das DPIL-Modell der organisatorischen Zuweisungsregeln des klinischen Prozesses (*Log 2*). Zu beachten ist, dass die Ereignisse des Logs in niederländischer Sprache vorliegen.

Listing 32: Organisatorische Zuweisungsregeln des klinischen Prozesses

```
%Identitaeten
use identity ObstetricsGynaecologyClinic
use identity Pathology
use identity Radiology
use identity Nursingward
use identity GeneralLabClinicalChemistry
use identity MedicalMicrobiology
use identity SpecialLabRadiology
use identity Anesthesiologyclinic
```

```
use identity NuclearMedicine
use identity Operatingrooms
use identity Recoveryroom/highcare
use identity Cardiovascularclinics
use identity DietStudies
use identity Endoscopy
use identity DayCentre-ward
use identity DayCentre-treatment
use identity PharmacyLaboratory
use identity Painclinic
use identity Radiotherapy
use identity InternalSpecialismsclinic

process Klinik_Log2 {

%Organisatorische Perspektive
ensure group(verlosk-gynaeckortekaartkosten-out, ObstetricsGynaecologyClinic)
ensure group(coupeterinzage, Pathology)
ensure group(thorax, Radiology)
ensure group(1econsultpoliklinisch, ObstetricsGynaecologyClinic)
ensure group(administratieftarief-eerstepol, ObstetricsGynaecologyClinic)
ensure group(ligdagen-allespecbehkinderg-reval, Nursingward)
ensure group(aannamelaboratoriumonderzoek, GeneralLabClinicalChemistry)
ensure group(sediment-spoed, GeneralLabClinicalChemistry)
ensure group(bilirubine-geconjugeerd, GeneralLabClinicalChemistry)
ensure group(bilirubine-totaal, GeneralLabClinicalChemistry)
ensure group(glucose, GeneralLabClinicalChemistry)
ensure group(ureum, GeneralLabClinicalChemistry)
ensure group(hemoglobinefoto-elektrisch, GeneralLabClinicalChemistry)
ensure group(creatinine, GeneralLabClinicalChemistry)
ensure group(alkalischefosfatase-kinetisch-, GeneralLabClinicalChemistry)
ensure group(natriumvlamfotometrisch, GeneralLabClinicalChemistry)
ensure group(kaliumpotentiometrisch, GeneralLabClinicalChemistry)
ensure group(sgot-asatkinetisch, GeneralLabClinicalChemistry)
ensure group(sgpt-alatkinetisch, GeneralLabClinicalChemistry)
ensure group(melkzuurdehydrogenase-ldh-kinetisch, GeneralLabClinicalChemistry)
ensure group(bacteriologischonderzoekmetkweek-nie, MedicalMicrobiology)
ensure group(bloedgroepaboenrhesusfactor, GeneralLabClinicalChemistry)
ensure group(rhesusfactord-centrifugeermethode-e, GeneralLabClinicalChemistry)
ensure group(leukocytentellenelektronisch, GeneralLabClinicalChemistry)
ensure group(trombocytentellen-elektronisch, GeneralLabClinicalChemistry)
ensure group(gammaglutamyltranspeptidase, GeneralLabClinicalChemistry)
ensure group(calcium, GeneralLabClinicalChemistry)
ensure group(urineonderzoek-kwalitatief, GeneralLabClinicalChemistry)
ensure group(albumine, GeneralLabClinicalChemistry)
ensure group(screeningantistoffenerytrocyten, GeneralLabClinicalChemistry)
ensure group(ordertarief, GeneralLabClinicalChemistry)
ensure group(mribekken, Radiology)
ensure group(190021klinischeopnamea002, Nursingward)
ensure group(190205klasse3ba205, Nursingward)
ensure group(telefonischconsult, ObstetricsGynaecologyClinic)
```

```
ensure group(lymfeklier-scintsentinelnodedynamis, NuclearMedicine)
ensure group(lymfeklier-scintsentinelnodevervolg, NuclearMedicine)
ensure group(lymfeklier-scintsentinelnodemetpro, NuclearMedicine)
ensure group(immunopathologischonderzoek, Pathology)
ensure group(vriescoupe, Pathology)
ensure group(cytologischonderzoek-ectocervix-, Pathology)
ensure group(histologischonderzoek-groteresectiep, Pathology)
ensure group(glucose-spoed, GeneralLabClinicalChemistry)
ensure group(methemoglobine-sulfhemoglobineelk, GeneralLabClinicalChemistry)
ensure group(bicarbonaat, GeneralLabClinicalChemistry)
ensure group(calcium-spoed, GeneralLabClinicalChemistry)
ensure group(co-hbkwn, GeneralLabClinicalChemistry)
ensure group(natrium-vlamfotometrisch-spoed, GeneralLabClinicalChemistry)
ensure group(kaliumvlamfotometrisch-spoed, GeneralLabClinicalChemistry)
ensure group(haemoglobinefoto-electrisch-spoed, GeneralLabClinicalChemistry)
ensure group(actueleph-pco2-standbicarbonaat, GeneralLabClinicalChemistry)
ensure group(melkzuurenzymatisch-spoed, GeneralLabClinicalChemistry)
ensure group(o2-saturatie, GeneralLabClinicalChemistry)
ensure group(sediment, GeneralLabClinicalChemistry)
advise group(verlosk-gynaecaanvkaartkosten-out, ObstetricsGynaecologyClinic)
ensure group(vervolgconsultpoliklinisch, ObstetricsGynaecologyClinic)
ensure group(histologischonderzoek-bioptennno, Pathology)
ensure group(cefalinetijd-coagulatie, GeneralLabClinicalChemistry)
ensure group(protrombinetijd, GeneralLabClinicalChemistry)
ensure group(vagina-toucheronderanesthesie, Operatingrooms)
ensure group(vulva-vulvectomie-liesblok, Operatingrooms)
ensure group(resistentiebepalingen-5bepalingen, MedicalMicrobiology)
ensure group(hepatitis-bsurfaceantigeenconfirmatie, MedicalMicrobiology)
ensure group(crpc-reactiefproteine, GeneralLabClinicalChemistry)
ensure group(leucocytentellen-electronisch-spoed, GeneralLabClinicalChemistry)
ensure group(creatinine-spoed, GeneralLabClinicalChemistry)
ensure group(differentieletellingautomatisch, GeneralLabClinicalChemistry)
ensure group(190101bovenregtoesla101, Nursingward)
ensure group(mriabdomen, Radiology)
ensure group(trombocytentellen-spoed, GeneralLabClinicalChemistry)
ensure group(hematocrietmbvcentrifuge, GeneralLabClinicalChemistry)
ensure group(vulva-ruimelokaleexcisievana, Operatingrooms)
ensure group(kruisproefvolledig-driemethoden-, GeneralLabClinicalChemistry)
ensure group(squamouscellcarcinomambveia, PharmacyLaboratory)
ensure group(gefiltreerderytrocytenconcentraat, GeneralLabClinicalChemistry)
}
```

Listing 33 zeigt das DPIL-Modell der verhaltensorientierten Perspektive des klinischen Prozesses (*Log 3*). Zu beachten ist wiederum, dass die Ereignisse des Logs in niederländischer Sprache vorliegen.

Listing 33: Verhaltensorientierte Perspektive des klinischen Prozesses

```
process Klinik_Log3 {

%Verhaltensorientierte Perspektive
ensure sequence(aannamelaboratoriumonderzoek, sediment-spoed)
```

```
ensure sequence(aannamelaboratoriumonderzoek, bilirubine-geconjugeerd)
ensure sequence(aannamelaboratoriumonderzoek, glucose)
ensure sequence(aannamelaboratoriumonderzoek, hemoglobinefoto-elektrisch)
ensure sequence(aannamelaboratoriumonderzoek, creatinine)
ensure sequence(aannamelaboratoriumonderzoek, bloedgroepaboenrhesusfactor)
ensure sequence(aannamelaboratoriumonderzoek, natrium-vlamfotometrisch-spoed)
ensure sequence(aannamelaboratoriumonderzoek, kaliumvlamfotometrisch-spoed)
ensure sequence(aannamelaboratoriumonderzoek, creatinine-spoed)
ensure sequence(bilirubine-geconjugeerd, bilirubine-totaal)
ensure sequence(bilirubine-totaal, alkalischefosfatase-kinetisch-)
ensure sequence(glucose, ureum)
ensure sequence(ureum, alkalischefosfatase-kinetisch-)
ensure sequence(ureum, calcium)
ensure sequence(hemoglobinefoto-elektrisch, alkalischefosfatase-kinetisch-)
ensure sequence(hemoglobinefoto-elektrisch, natriumvlamfotometrisch)
ensure sequence(hemoglobinefoto-elektrisch, sgot-asatkinetisch)
ensure sequence(hemoglobinefoto-elektrisch, urineonderzoek-kwalitatief)
ensure sequence(creatinine, alkalischefosfatase-kinetisch-)
ensure sequence(creatinine, natriumvlamfotometrisch)
ensure sequence(alkalischefosfatase-kinetisch-, melkzuurdehydrogenase-ldh-kinetisch)
ensure sequence(natriumvlamfotometrisch, kaliumpotentiometrisch)
ensure sequence(kaliumpotentiometrisch, sgpt-alatkinetisch)
ensure sequence(kaliumpotentiometrisch, trombocytentellen-elektronisch)
ensure sequence(kaliumpotentiometrisch, hematocrietmbvcentrifuge)
ensure sequence(sgot-asatkinetisch, sgpt-alatkinetisch)
ensure sequence(sgot-asatkinetisch, leukocytentellenelektronisch)
ensure sequence(sgpt-alatkinetisch, melkzuurdehydrogenase-ldh-kinetisch)
ensure sequence(melkzuurdehydrogenase-ldh-kinetisch, gammaglutamyltranspeptidase)
ensure sequence(bloedgroepaboenrhesusfactor, rhesusfactord-centrifugeermethode-e)
ensure sequence(rhesusfactord-centrifugeermethode-e, leukocytentellenelektronisch)
ensure sequence(rhesusfactord-centrifugeermethode-e, trombocytentellen-elektronisch)
ensure sequence(rhesusfactord-centrifugeermethode-e, urineonderzoek-kwalitatief)
ensure sequence(rhesusfactord-centrifugeermethode-e, screeningantistoffenerytrocyten)
ensure sequence(rhesusfactord-centrifugeermethode-e, haemoglobinefoto-electrisch-spoed)
ensure sequence(rhesusfactord-centrifugeermethode-e, differentieletellingautomatisch)
ensure sequence(rhesusfactord-centrifugeermethode-e, hematocrietmbvcentrifuge)
ensure sequence(rhesusfactord-centrifugeermethode-e, kruisproefvolledig-driemethoden-)
ensure sequence(leukocytentellenelektronisch, gammaglutamyltranspeptidase)
ensure sequence(leukocytentellenelektronisch, albumine)
ensure sequence(leukocytentellenelektronisch, crpc-reactiefproteine)
ensure sequence(trombocytentellen-elektronisch, gammaglutamyltranspeptidase)
ensure sequence(trombocytentellen-elektronisch, calcium)
ensure sequence(trombocytentellen-elektronisch, cefalinetijd-coagulatie)
ensure sequence(gammaglutamyltranspeptidase, sediment)
ensure sequence(calcium, sediment)
ensure sequence(albumine, sediment)
ensure sequence(screeningantistoffenerytrocyten, glucose-spoed)
ensure sequence(screeningantistoffenerytrocyten, calcium-spoed)
ensure sequence(screeningantistoffenerytrocyten, o2-saturatie)
ensure sequence(screeningantistoffenerytrocyten, sediment)
ensure sequence(screeningantistoffenerytrocyten, protrombinetijd)
```

```
ensure sequence(screeningantistoffenerytrocyten, crpc-reactiefproteine)
ensure sequence(screeningantistoffenerytrocyten, gefiltreerderytrocytenconcentraat)
ensure sequence(ordertarief, glucose-spoed)
ensure sequence(ordertarief, calcium-spoed)
ensure sequence(ordertarief, natrium-vlamfotometrisch-spoed)
ensure sequence(ordertarief, kaliumvlamfotometrisch-spoed)
ensure sequence(ordertarief, o2-saturatie)
ensure sequence(ordertarief, sediment)
ensure sequence(ordertarief, crpc-reactiefproteine)
ensure sequence(ordertarief, creatinine-spoed)
ensure sequence(ordertarief, kruisproefvolledig-driemethoden-)
ensure sequence(glucose-spoed, methemoglobine-sulfhemoglobineelk)
ensure sequence(glucose-spoed, melkzuurenzymatisch-spoed)
ensure sequence(methemoglobine-sulfhemoglobineelk, bicarbonaat)
ensure sequence(bicarbonaat, co-hbkwn)
ensure sequence(calcium-spoed, co-hbkwn)
ensure sequence(calcium-spoed, melkzuurenzymatisch-spoed)
ensure sequence(co-hbkwn, actueleph-pco2-standbicarbonaat)
ensure sequence(natrium-vlamfotometrisch-spoed, actueleph-pco2-standbicarbonaat)
ensure sequence(natrium-vlamfotometrisch-spoed, melkzuurenzymatisch-spoed)
ensure sequence(kaliumvlamfotometrisch-spoed, actueleph-pco2-standbicarbonaat)
ensure sequence(kaliumvlamfotometrisch-spoed, melkzuurenzymatisch-spoed)
ensure sequence(haemoglobinefoto-electrisch-spoed, actueleph-pco2-standbicarbonaat)
ensure sequence(haemoglobinefoto-electrisch-spoed, melkzuurenzymatisch-spoed)
ensure sequence(haemoglobinefoto-electrisch-spoed, leucocytentellen-electronisch-spoed)
ensure sequence(haemoglobinefoto-electrisch-spoed, trombocytentellen-spoed)
ensure sequence(cefalinetijd-coagulatie, protrombinetijd)
ensure sequence(kruisproefvolledig-driemethoden-, gefiltreerderytrocytenconcentraat)
}
```

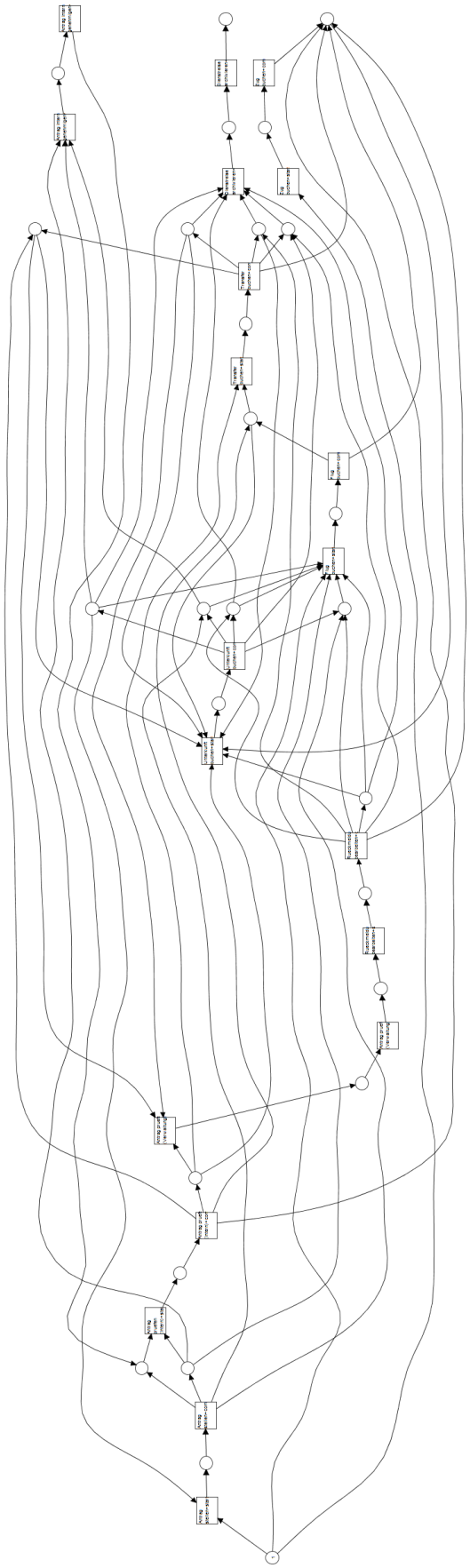


Abb. 49: Extrahiertes Prozessmodell des α-Algorithmus

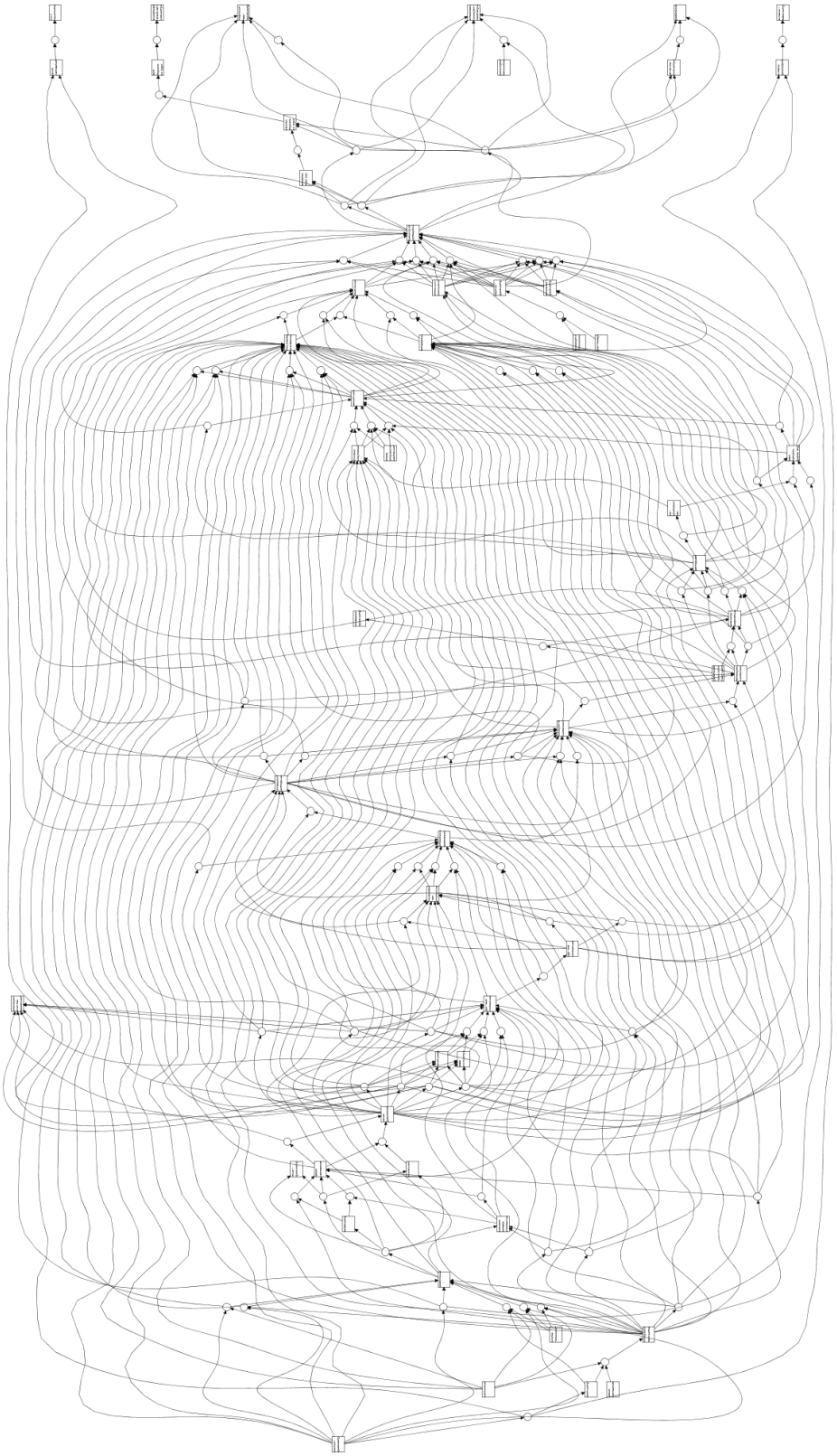


Abb. 50: Extrahiertes Modell des klinischen Prozesses durch den α-Algorithmus

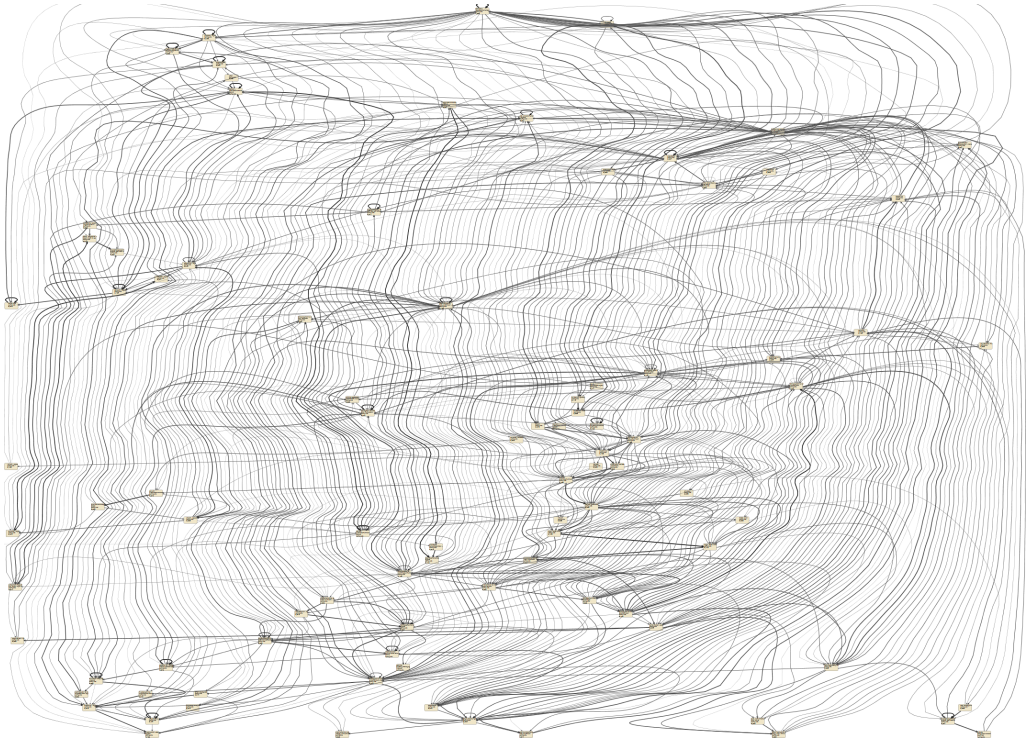


Abb. 51: Extrahiertes Modell des klinischen Prozesses durch den FuzzyMiner (ohne Filterung)

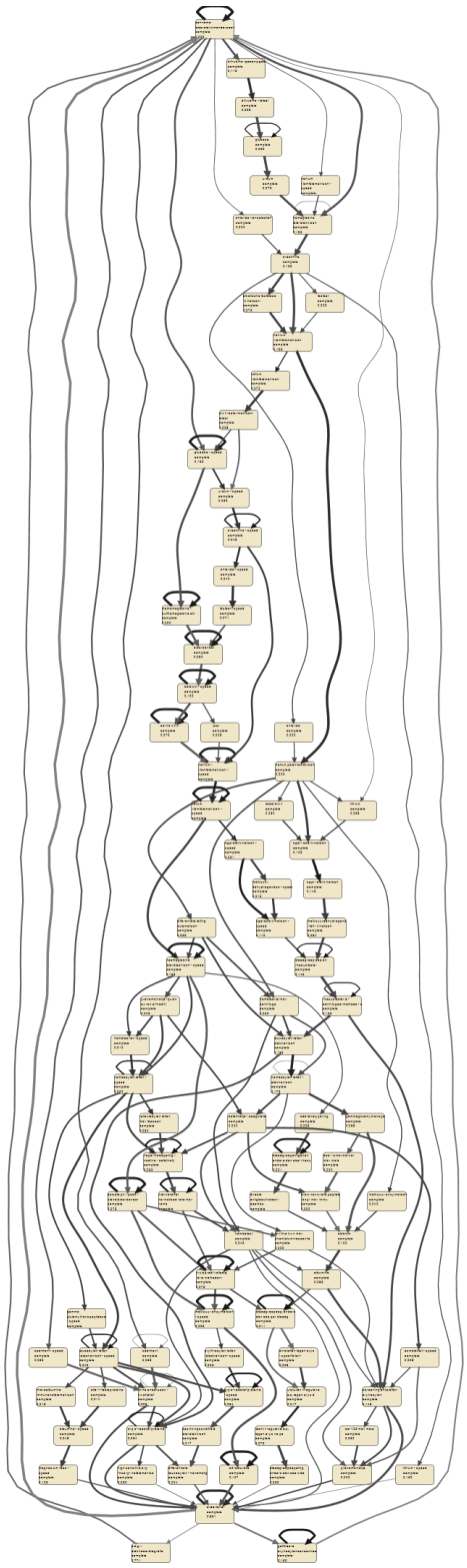


Abb. 52: Extrahiertes Modell des klinischen Prozesses durch den FuzzyMiner (mit Filterung)

LITERATUR

[1] W. M. van der Aalst, H. De Beer und B. F. van Dongen, *Process mining and verification of properties: An approach based on temporal logic*. Springer, 2005.

[2] W. van der Aalst, "Desire Lines in Big Data", in *Encyclopedia of Social Network Analysis and Mining*, R. A. Rokne und J., Hrsg., Berlin: Springer Berlin / Heidelberg, 2014.

[3] W. van der Aalst, H. de Beer und B. van Dongen, "Process Mining and Verification of Properties: An Approach Based on Temporal Logic", in *On the Move to Meaningful Internet Systems (OTM Workshops)*, Springer, 2005, S. 130–147.

[4] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski und A. Barros, "Workflow Patterns", *Distributed and parallel databases*, Bd. 14, Nr. 1, S. 5–51, 2003.

[5] W. van der Aalst und M. Song, "Mining Social Networks: Uncovering interaction patterns in business processes", in *Business Process Management*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, S. 244–260.

[6] W. van der Aalst und T. Weijters, "Process mining: a research agenda", *Computers in Industry*, Bd. 53, Nr. 3, S. 231–244, Apr. 2004.

[7] W. van der Aalst, T. Weijters und L. Maruster, "Workflow mining: Discovering process models from event logs", *IEEE Transactions on Knowledge and Data Engineering*, Bd. 16, Nr. 9, S. 1128–1142, 2004.

[8] W. van der Aalst, M. Weske und D. Grünbauer, "Case handling: a new paradigm for business process support", *Data & Knowledge Engineering*, Bd. 53, Nr. 2, S. 129–162, Mai 2005.

[9] W. van der Aalst, "Process Discovery: Capturing the Invisible", *Computational Intelligence Magazine, IEEE*, Bd. 5, Nr. 1, S. 28–41, 2010.

[10] W. van der Aalst, "Process Mining Manifesto", in *Business Process Management Workshops*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 169–194.

[11] W. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Berlin Heidelberg, 2011.

[12] A. Adriansyah, B. F. van Dongen und W. M. van der Aalst, "Conformance checking using cost-based fitness analysis", in *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, IEEE, 2011, S. 55–64.

[13] R. Agrawal, D. Gunopulos und F. Leymann, *Mining process models from workflow logs*. Springer, 1998.

[14] R. Agrawal, R. Srikant u. a., "Fast algorithms for mining association rules", in *Proc. 20th int. conf. very large data bases, VLDB*, Bd. 1215, 1994, S. 487–499.

[15] R. Agrawal und R. Srikant, "Mining Sequential Patterns", in *Proceedings of the Eleventh International Conference on Data Engineering*, 1995, S. 3–14.

[16] A. V. Aho, M. R. Garey und J. D. Ullman, "The transitive reduction of a directed graph", *SIAM Journal on Computing*, Bd. 1, Nr. 2, S. 131–137, 1972.

[17] R. H. von Alan, S. T. March, J. Park und S. Ram, "Design science in information systems research", *MIS quarterly*, Bd. 28, Nr. 1, S. 75–105, 2004.

[18] R. Angles und C. Gutierrez, "The expressive power of SPARQL", in *The Semantic Web-ISWC 2008*, 2008.

[19] A. Awad, G. Decker und M. Weske, "Efficient compliance checking using bpmn-q and temporal logic", in *Business Process Management*, Springer, 2008, S. 326–341.

[20] I. Barba, B. Weber und C. D. Valle, "Supporting the optimized execution of business processes through recommendations", in *Business Process Management Workshops*, Springer Berlin Heidelberg, 2012, S. 135–140.

[21] M. H. Baumann, M. Baumann, S. Schönig und S. Jablonski, "Resource-Aware Process Model Similarity Matching", in *1st Workshop on Resource Management in Service-Oriented Computing (RMSOC)*, Springer, 2014.

[22] M. H. Baumann, M. Baumann, S. Schönig und S. Jablonski, "Towards Multi-Perspective Process Model Similarity Matching", in *10th International Workshop on Enterprise & Organizational Modeling and Simulation (EOMAS 2014), in conjunction with CAiSE'14*, Thessaloniki, Greece: Springer LNBIP, 2014, S. 1–17.

[23] M. Baumann, M. H. Baumann, S. Schönig und S. Jablonski, "Enhancing Feasibility of Human-driven Processes by Transforming Process Models to Process Checklists", in *15th Working Conference of Business Process Modeling, Development, and Support (BPMDS 2014)*, Springer Berlin Heidelberg, 2014.

[24] A. Baumgrass, S. Schefer-Wenzl und M. Strembeck, "Deriving process-related rbac models from process execution histories", in *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*, IEEE, 2012, S. 421–426.

[25] A. Baumgrass und M. Strembeck, "Bridging the gap between role mining and role engineering via migration guides", *Information Security Technical Report*, Bd. 17, Nr. 4, S. 148–172, 2013.

[26] R. Bergenthum, J. Desel, R. Lorenz und S. Mauser, "Process mining based on regions of languages", in *Business Process Management*, Springer, 2007, S. 375–383.

[27] J. C. Bose, F. M. Maggi und W. van der Aalst, "Enhancing Declare Maps Based on Event Correlations", in *Business Process Management*, Springer Berlin Heidelberg, 2013, S. 97–112.

[28] R. J. C. Bose und W. M. van der Aalst, "Analysis of patient treatment procedures.", in *Business Process Management Workshops (1)*, 2011, S. 165–166.

[29] J. Broekstra, A. Kampman und F. Van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF schema", in *The Semantic Web—ISWC 2002*, Springer, 2002, S. 54–68.

[30] C. Bussler, *Organisationsverwaltung in Workflow-Management-Systemen*. Dt. Univ.-Verlag, 1998.

[31] C. Cabanillas, D. Knuplesch, M. Resinas, M. Reichert, J. Mendling und A. Ruiz-Cortés, "RALph: A Graphical Notation for Resource Assignments in Business Processes", in *CAiSE*, 2015, In press.

[32] C. Cabanillas, M. Resinas und A. R. Cortés, "RAL: A high-level user-oriented resource assignment language for business processes", in *Business Process Management Workshops*, 2011, S. 50–61.

[33] F. Caron, J. Vanthienen und B. Baesens, "A comprehensive investigation of the applicability of process mining techniques for enterprise risk management", *Computers in Industry*, Bd. 64, Nr. 4, S. 464–475, 2013.

[34] F. Caron, J. Vanthienen und B. Baesens, "Comprehensive rule-based compliance checking and risk management with process mining", *Decision Support Systems*, Bd. 54, Nr. 3, S. 1357–1369, 2013.

[35] F. Caron, J. Vanthienen und B. Baesens, "Advances in Rule-Based Process Mining: Applications for Enterprise Risk Management and Auditing", 2013.

[36] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne und K. Wilkinson, "Jena: implementing the semantic web recommendations", in *World Wide Web 2004*, ACM, 2004, S. 74–83.

[37] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi und S. Storari, "Exploiting inductive logic programming techniques for declarative process mining", *Transactions on Petri Nets and Other Models of Concurrency*, S. 278–295, 2009.

[38] C. D. Ciccio und M. Mecella, "MINERful, a mining algorithm for declarative process constraints in MailOfMine", *Department of Computer and System Sciences Antonio Ruberti Technical Reports*, Bd. 4, Nr. 3, 2012.

[39] J. E. Cook und A. L. Wolf, "Discovering models of software processes from event-based data", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Bd. 7, Nr. 3, S. 215–249, 1998.

[40] B. Curtis, M. Kellner und J. Over, "Process modeling", *Communications of the ACM*, Nr. 9, S. 75–90, 1992.

[41] A. Datta, "Automating the discovery of as-is business process models: probabilistic and algorithmic approaches", *Information Systems Research*, Bd. 9, Nr. 3, S. 275–301, 1998.

[42] T. Davenport und J. Short, "The New Industrial Engineering: Information Technology and Business Process Redesign", *Sloan Management Review*, Bd. 31, Nr. 4, S. 11–27, 1990.

[43] M. De Leoni, F. M. Maggi und W. M. van der Aalst, "Aligning event logs and declarative process models for conformance checking", in *Business Process Management*, Springer, 2012, S. 82–97.

[44] W. De Roover, F. Caron und J. Vanthienen, "A prototype tool for the event-driven enforcement of sbvr business rules", in *Business Process Management Workshops*, Springer, 2012, S. 446–457.

[45] C. Di Ciccio und M. Mecella, "A two-step fast algorithm for the automated discovery of declarative workflows", in *Computational Intelligence and Data Mining*, 2013, S. 135–142.

[46] B. F. van Dongen, A. A. De Medeiros und L. Wen, "Process mining: overview and outlook of petri net discovery algorithms", in *Transactions on Petri Nets and Other Models of Concurrency II*, Springer, 2009, S. 225–242.

[47] M. Dumas, M. La Rosa, J. Mendling und H. Reijers, *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[48] D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich und S. Zugal, "Declarative versus imperative process modeling languages: The issue of understandability", in *Enterprise, Business-Process and Information Systems Modeling*, Springer Berlin Heidelberg, 2009, S. 353–366.

[49] M. Färber, *Prozessorientiertes Qualitätsmanagement: Ein Konzept zur Implementierung*. Dissertation, Universität Bayreuth, Gabler Verlag, 2010.

[50] U. Fayyad, G. Piatetsky-Shapiro und P. Smyth, "From Data Mining to Knowledge Discovery in Databases", *AI magazine*, S. 37–54, 1996.

[51] C. Forgy, "Rete: a fast algorithm for the many pattern/many object pattern match problem", *Artificial intelligence*, Bd. 19, Nr. 1, S. 17–37, 1982.

[52] N. S. Glance, D. S. Pagani und R. Pareschi, "Generalized process structure grammars gpsg for flexible representations of work", in *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, ACM, 1996, S. 180–189.

[53] S. Goedertier und J. Vanthienen, "Declarative process modeling with business vocabulary and business rules", in *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, Springer, 2007, S. 603–612.

[54] E. M. Gold, "Language identification in the limit", *Information and control*, Bd. 10, Nr. 5, S. 447–474, 1967.

[55] C. Görg, M. Pohl, E. Qeli und K. Xu, "Visual representations", in *Human-Centered Visualization Environments*, Springer, 2007, S. 163–230.

[56] N. Gronau und E. Weber, "Management of knowledge intensive business processes", in *Business Process Management*, Springer, 2004, S. 163–178.

[57] C. Günther, *Process mining in flexible environments*. Eindhoven, 2009.

[58] C. Günther und W. van der Aalst, "Fuzzy mining–adaptive process simplification based on multi-perspective metrics", in *Business Process Management*, Springer Berlin Heidelberg, 2007, S. 328–343.

[59] C. Günther und H. Verbeek, *XES-Standard Definition*, 2014.

[60] C. Günther, S. Schönig und S. Jablonski, "Dynamic Guidance Enhancement in Workflow Management Systems", in *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012)*, New York, New York, USA: ACM Press, 2012, S. 1717.

[61] S. Harris und A. Seaborne, "SPARQL 1.1 Query Language", *W3C Recommendation*, Bd. 21, 2013.

[62] J. Herbst, "A machine learning approach to workflow management", in *Machine Learning: ECML 2000*, Springer, 2000, S. 183–194.

[63] T. Hildebrandt, R. R. Mukkamala, T. Slaats und F. Zanitti, "Contracts for cross-organizational workflows as timed dynamic condition response graphs", *The Journal of Logic and Algebraic Programming*, Bd. 82, Nr. 5, S. 164–185, 2013.

[64] R. Hull, E. Damaggio, R. De Masellis, F. Fournier, M. Gupta, F. T. Heath III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam u. a., "Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events", in *Proceedings of the 5th ACM international conference on Distributed event-based system*, ACM, 2011, S. 51–62.

[65] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. T. Heath III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya u. a., "Introducing the guard-stage-milestone approach for specifying business entity lifecycles", in *Web services and formal methods*, Springer, 2011, S. 1–24.

[66] M. Igler, *ESProNa - Eine Constraintsprache zur multimodalen Prozessmodellierung und navigationsgestützten Ausführung*. Bayreuth: Dissertation, Universität Bayreuth, 2012.

[67] S. Jablonski, "MOBILE: A modular workflow model and architecture", in *Proceedings of the 4th International Working Conference on Dynamic Modelling and Information Systems*, Delft University Press, 1994.

[68] S. Jablonski und C. Bussler, *Workflow Management: Modeling Concepts, Architecture and Implementation*. International Thomson Publishing Services, 1996.

[69] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use*. Springer Science & Business Media, 1997, Bd. 1.

[70] T. Jin, J. Wang und L. Wen, "Organizational modeling from event logs", in *Grid and Cooperative Computing, 2007. GCC 2007. Sixth International Conference on*, IEEE, 2007, S. 670–675.

[71] E. Lamma, P. Mello, M. Montali, F. Riguzzi und S. Storari, "Inducing declarative logic-based models from labeled traces", in *Business Process Management*, Springer, 2007, S. 344–359.

[72] O. Lassila und R. R. Swick, "Resource Description Framework (RDF) Model and Syntax Specification", 1999.

[73] M. Leitner, A. Baumgrass, S. Schefer-Wenzl, S. Rinderle-Ma und M. Strembeck, "A case study on the suitability of process mining to produce current-state rbac models", in *Business Process Management Workshops*, 2013, S. 719–724.

[74] R. Lenz, M. Peleg und M. Reichert, "Healthcare process support: achievements, challenges, current research", *International Journal of Knowledge-Based Organizations (IJKBO)*, Bd. 2, Nr. 4, 2012.

[75] D. Liu, J. Wang, S. C. Chan, J. Sun und L. Zhang, "Modeling workflow processes with colored petri nets", *Computers in Industry*, Bd. 49, Nr. 3, S. 267–281, 2002.

[76] L. T. Ly, S. Rinderle, P. Dadam und M. Reichert, "Mining staff assignment rules from event-based data", in *Business process management workshops*, Springer, 2006, S. 177–190.

[77] F. M. Maggi, "Declarative Process Mining with the Declare Component of ProM", in *BPM Demos, CEUR Workshop Proceedings, CEUR-WS.org*, 2013.

[78] F. M. Maggi, J. C. Bose und W. M. van der Aalst, "A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps", in *Advanced Information Systems Engineering*, 2013, S. 433–448.

[79] F. M. Maggi, J. C. Bose und W. van der Aalst, "Efficient Discovery of Understandable Declarative Process Models from Event Logs", in *Advanced Information Systems Engineering*, Springer Berlin Heidelberg, 2012, S. 270–285.

[80] F. M. Maggi und M. Dumas, "Discovering Data-Aware Declarative Process Models from Event Logs", in *Business Process Management*, 2013, S. 1–16.

[81] F. M. Maggi, A. Mooij und W. van der Aalst, "User-Guided Discovery of Declarative Process Models", in *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Ieee, Apr. 2011, S. 192–199.

[82] F. M. Maggi, T. Slaats und H. A. Reijers, "The automated discovery of hybrid processes", in *Business Process Management*, Springer, 2014, S. 392–399.

[83] H. D. Man, "Case management: Cordys approach", *BPTrends*, Nr. February, S. 1–13, 2009.

[84] H. Mannila, H. Toivonen und I. Verkamo, "Discovery of frequent episodes in event sequences", *Data Mining and Knowledge Discovery*, Bd. 1, Nr. 3, S. 259–289, 1997.

[85] R. Mans, H. Schonenberg, M. Song, W. van der Aalst und P. Bakker, "Application of process mining in healthcare–a case study in a dutch hospital", in *Biomedical Engineering Systems and Technologies*, 2009, S. 425–438.

[86] A. K. A. de Medeiros, T. Weijters und W. van der Aalst, "Genetic process mining: an experimental evaluation", *Data Mining and Knowledge Discovery*, Bd. 14, Nr. 2, S. 245–304, Jan. 2007.

[87] S. Meerkamm, *Ein Rahmenwerk für das Prozessdesign zur Identifikation, Klassifikation und Umsetzung von Anforderungen*. Dissertation, Universität Bayreuth, 2012.

[88] M. Montali, M. Pesic, W. M. van der Aalst, F. Chesani, P. Mello und S. Storari, "Declarative Specification and Verification of Service Choreographies", *ACM Transactions on the Web (TWEB)*, Bd. 4, Nr. 1, S. 3, 2010.

[89] M. Montali, P. Torroni, M. Alberti, F. Chesani, M. Gavanelli, E. Lamma und P. Mello, "Verification from declarative specifications using logic programming", in *Logic Programming*, Springer, 2008, S. 440–454.

[90] R. R. Mukkamala, "A formal model for declarative workflows: dynamic condition response graphs", Diss., University of Copenhagen, 2012.

[91] C. Myhill, "Commercial success by looking for desire lines", in *Computer Human Interaction*, Springer Berlin Heidelberg, 2004, S. 293–304.

[92] J. Nakatumba, *Resource-aware business process management: analysis and support*. Eindhoven: Technische Universiteit Eindhoven, 2013.

[93] J. Nakatumba und W. van der Aalst, "Analyzing resource behavior using process mining", in *Business Process Management Workshops*, Springer, 2010, S. 69–80.

[94] A. Nonnengart und H. J. Ohlbach, "Modal- und Temporal-Logik", in *Deduktionssysteme - Automatisierung des logischen Denkens*, Oldenbourg, 1992, S. 239–284.

[95] Object Management Group, *Decision Model and Notation (DMN) - Beta 1*, 2014.

[96] Object Management Group (OMG), *Business Process Model and Notation (BPMN) - Version 2.0*, 2011.

[97] Object Management Group (OMG), *Case Management Model and Notation (CMMN), Version 1.0*, 2014.

[98] Object Management Group (OMG), *Semantics of Business Vocabulary and Business Rules (SBVR)*, 2013.

[99] J. Pérez, M. Arenas und C. Gutierrez, "Semantics and Complexity of SPARQL", in *The Semantic Web-ISWC 2006*, 2006. arXiv: 0605124v1 [arXiv:cs].

[100] R. Pérez-Castillo, B. Weber, J. Pinggera, S. Zugal, I. G.-R. de Guzmán und M. Piattini, "Generating event logs from non-process-aware systems enabling business process mining", *Enterprise Information Systems*, Bd. 5, Nr. 3, S. 301–335, Aug. 2011.

[101] M. Pesic, "Constraint-Based Workflow Management Systems: Shifting Control to Users", Diss., Eindhoven University of Technology, 2008.

[102] M. Pesic, H. Schonenberg und W. van der Aalst, "DECLARE: Full Support for Loosely-Structured Processes", *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, S. 287–287, Okt. 2007.

[103] M. Pesic und W. M. Van der Aalst, "A declarative approach for flexible business processes management", in *Business Process Management Workshops*, Springer, 2006, S. 169–180.

[104] G. Piatetsky-Shapiro, "Knowledge Discovery in Real Databases : A Report on the IJCAI-89 Workshop", *AI Magazine*, Bd. 11, Nr. 5, 1990.

[105] G. Regev und A. Wegmann, "A Regulation-Based View on Business Process and Supporting System Flexibility", in *Proceedings of the CAiSE*, 2005, S. 91–98.

[106] M. Reichert und B. Weber, *Enabling Flexibility in Process-aware Information Systems: Challenges*. Springer Berlin Heidelberg, 2012.

[107] S. Rinderle-Ma und W. M. van der Aalst, "Life-cycle support for staff assignment rules in process-aware information systems", 2007.

[108] B. Roth, "Beispielgetriebene entwicklung domänenspezifischer modellierungssprachen", Diss., Universität Bayreuth, 2014.

[109] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. Weijters und W. M. van der Aalst, "The need for a process mining evaluation framework in research and practice", in *Business Process Management Workshops*, Springer, 2008, S. 84–89.

[110] N. Russell, A. ter Hofstede und D. Edmond, "Workflow Data Patterns: Identification, representation and tool support", in *Conceptual Modeling–ER*, 2005, S. 353–368.

[111] N. Russell, A. ter Hofstede, D. Edmond und W. van der Aalst, "Workflow Resource Patterns: Identification, representation and tool support", in *Advanced Information Systems Engineering*, 2005.

[112] S. Schönig, C. Cabanillas, S. Jablonski und J. Mendling, "Mining the Organisational Perspective in Agile Business Processes", in *Enterprise, Business-Process and Information Systems Modeling*, Springer, 2015.

[113] S. Schönig, C. Günther und S. Jablonski, "Process Discovery and Guidance Applications of Manually Generated Logs", in *The Seventh International Conference on Internet Monitoring and Protection (ICIMP 2012)*, Stuttgart, Germany, 2012.

[114] S. Schönig und S. Jablonski, "Comparing Declarative Process Modelling Languages from the Organisational Perspective", in *4th International Workshop on Adaptive Case Management and other non-workflow approaches to BPM (AdaptiveCM 2015)*, Springer, 2015.

[115] S. Schönig, M. Seitz, C. Piesche, M. Zeising und S. Jablonski, "Process Observation as Support for Evolutionary Process Engineering", *International Journal On Advances in Systems and Measurements*, Bd. 5, Nr. 3, S. 188–202, 2012.

[116] S. Schönig und M. Zeising, "The DPIL Framework: Tool Support for Agile and Resource-Aware Business Processes", in *BPM 2015 Demos*, CEUR, 2015.

[117] S. Schönig, M. Zeising, F. Gillitzer und S. Jablonski, "Supporting Rule-based Process Mining by User-Guided Discovery of Resource-Aware Frequent Patterns", in *1st Workshop on Resource Management in Service-Oriented Computing (RMSOC)*, Springer, 2014.

[118] S. Schönig, M. Zeising und S. Jablonski, "Adapting Association Rule Mining to Discover Patterns of Collaboration in Process Logs", in *Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Ieee, 2012.

[119] S. Schönig, C. Günther, M. Zeising und S. Jablonski, "Discovering Cross-Perspective Semantic Definitions from Process Execution Logs", in *The Second International Conference on Business Intelligence and Technology (BUSTECH 2012)*, Nice, France, 2012.

[120] S. Schönig, M. Zeising und S. Jablonski, "Supporting Collaborative Work by Learning Process Models and Patterns from Cases", in *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2013.

[121] S. Schönig, M. Zeising und S. Jablonski, "Comprehensive Business Process Management through Observation and Navigation", in *Working Conference on the Practice of Enterprise Modeling (PoEM)*, 2013.

[122] S. Schönig, M. Zeising und S. Jablonski, "Towards Location-Aware Declarative Business Process Management", in *6th Workshop on Applications of Knowledge-Based Technologies in Business (AKTB 2014), in conjunction with BIS 2014*, Springer LNBIP, 2014.

[123] M. Seitz, S. Schönig und S. Jablonski, "A Framework for Reasonable Support of Process Compliance Management", in *5th Workshop on Business and IT Alignment (BITA 2014) in conjunction with BIS 2014*, Larnaca, Cyprus: Springer LNBIP, 2014.

[124] Software AG, "ARIS Method - ARIS Platform Version 7.2 - Service Release 3", Darmstadt, Techn. Ber., 2012.

[125] M. Song und W. van der Aalst, "Towards comprehensive support for organizational mining", *Decision Support Systems*, 2008.

[126] K. Swenson, *Flipping the process life cycle*, http://social-biz.org/2011/12/04/flipping-the-process-life-cycle/, Dez. 2011.

[127] K. Swenson, N. Palmer und B. Silver, *Taming the Unpredictable*, L. Fischer, Hrsg. Future Strategies, 2011.

[128] K. Swenson, *Mastering the Unpredictable: The Adaptive Case Management Revolution*. Meghan-Kiffer Press, 2010.

[129] J. D. Team, *Jboss drools documentation - chapter 7: rule language reference*, 2013.

[130] R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam und P. Sukaviriya, "Declarative business artifact centric modeling of decision and knowledge intensive business processes", in *Enterprise Distributed Object Computing Conference (EDOC)*, 2011, S. 151–160.

[131] R. Vaculin, R. Hull, M. Vukovic, T. Heath, N. Mills und Y. Sun, "Supporting Collaborative Decision Processes", in *IEEE International Conference on Services Computing*, Ieee, Juni 2013, S. 651–658.

[132] W. Van der Aalst, "The application of petri nets to workflow management", *Journal of circuits, systems, and computers*, Bd. 8, Nr. 01, S. 21–66, 1998.

[133] W. M. Van Der Aalst und M. Pesic, *DecSerFlow: Towards a truly declarative service flow language*. Springer, 2006.

[134] W. Van der Aalst, A. Adriansyah und B. van Dongen, "Replaying history on process models for conformance checking and performance analysis", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Bd. 2, Nr. 2, S. 182–192, 2012.

[135] D. Vanderhaeghen, P. Fettke und P. Loos, "Organisations- und Technologieoptionen des Geschäftsprozessmanagements aus der Perspektive des Web 2.0", *Wirtschaftsinformatik*, Bd. 52, Nr. 1, S. 17–32, Jan. 2010.

[136] E. Verbeek, J. Buijs, B. van Dongen und W. van der Aalst, "XES, xESame, and proM 6", in *Information Systems Evolution*, 2011, S. 60–75.

[137] T. Weijters und A. K. A. de Medeiros, "Process mining with the heuristics miner-algorithm", *BETA Working Paper Series, WP 166, Eindhoven University of Technology*, 2006.

[138] M. Weske, *Business process management: concepts, languages, architectures*. Springer Berlin Heidelberg, 2012.

[139] M. Westergaard, C. Stahl und H. Reijers, "UnconstrainedMiner: Efficient Discovery of Generalized Declarative Process Models", *BPM Center Report, No. BPM-13-28*, 2013.

[140] G. Williams, *Data Mining with Rattle and R: the art of excavating data for knowledge discovery*. Springer Berlin Heidelberg, 2011.

[141] M. Zeising, *Plattform zur integrierten IT-gestützten Ausführung von strikten und agilen Prozessen*. Dissertation, Universität Bayreuth, 2015.

[142] M. Zeising, S. Schönig und S. Jablonski, "Improving collaborative business process execution by traceability and expressiveness", in *Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Ieee, 2012.

[143] M. Zeising, S. Schönig und S. Jablonski, "Towards a Common Platform for the Support of Routine and Agile Business Processes", in *Proceedings of the 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2014.

[144] W. Zhao und X. Zhao, "Process mining from the organizational perspective", in *Foundations of Intelligent Systems*, Springer, 2014, S. 701–708.

[145] M. Zur Muehlen, M. Indulska und K. Kittel, "Towards integrated modeling of business processes and business rules", *ACIS 2008 Proceedings*, S. 108, 2008.